T H E S E

PRESENTEE

A L'UNIVERSITE DE PARIS VII

POUR L'OBTENTION

DU GRADE DE DOCTEUR INGENIEUR EN ELECTRONIQUE
(Traitement de l'Information)

par

Ernest BLANCA

CONTRIBUTION A LA CONCEPTION ASSISTEE PAR ORDINATEUR D'ENSEMBLES LOGIQUES

TRAITEMENT DES EXPRESSIONS BOOLEENNES

PROGRAMME CHOPIN

Soutenue le 1er juin 1973, devant la Commission d'Examen

MM.	A.	BLAQUIERE	Président
	P.	DEBRAINE	
	F.	BERTEIN	Examinateurs
Melle	H.	BESTOUGEFF	
M.	J.	POTTIER	Membre invité

T H E S E

PRESENTEE

A L'UNIVERSITE DE PARIS VII

POUR L'OBTENTION

DU GRADE DE DOCTEUR INGENIEUR EN ELECTRONIQUE
(Traitement de l'Information)

par

Ernest BLANCA

CONTRIBUTION A LA CONCEPTION ASSISTEE PAR ORDINATEUR D'ENSEMBLES LOGIQUES

TRAITEMENT DES EXPRESSIONS BOOLEENNES

PROGRAMME CHOPIN

Soutenue le 1er juin 1973, devant la Commission d'Examen

MM.	A.	BLAQUIERE	Président
	P.	DEBRAINE	
	F.	BERTEIN	Examinateurs
Melle	H.	BESTOUGEFF	
M.	J.	POTTIER	Membre invité

UNIV.
LEIDEN
BIBL.

A

Alain, François, Laurence, Xavier et Yves

avec mon affection.

J.

Digitized by the Internet Archive
in 2026

https://archive.org/details/IA41548016_0013

Monsieur A. BLAQUIERE, Professeur à la Faculté des Sciences de Paris, à qui je dois mes premiers pas en Mathématiques dites Supérieures, a bien voulu me faire l'honneur d'accepter la présidence de ce jury. Je le prie de trouver ici l'expression de ma respectueuse reconnaissance.

J'adresse mes plus vifs remerciements à Monsieur le Professeur F. BERTEIN, de la Faculté des Sciences de Paris-Sud (Orsay), dont l'intérêt pour la Conception Assistée par Ordinateur nous a permis d'en montrer une application à un groupe de ses étudiants et qui a bien voulu faire partie du jury.

Je remercie également Mademoiselle H. BESTOUGEFF de la Faculté des Sciences de Paris, d'avoir accepté d'être membre de ce jury.

Je prie Monsieur P. DEBRAINE, Adjoint au Directeur du Centre d'Etudes Nucléaires de Saclay, Professeur à l'Institut National des Sciences et Techniques Nucléaires, de trouver ici l'expression de ma profonde reconnaissance pour avoir bien voulu patronner ce travail, pour les conseils, les critiques et l'enseignement de valeur que j'en ai reçus, par l'accueil toujours aimable que j'ai trouvé auprès de lui.

Je me permets d'adresser mes remerciements à Monsieur J. DEBIESSE, Directeur de l'Institut National des Sciences et Techniques Nucléaires, à Monsieur M. DOIREAU, qui avait la responsabilité du Département d'Electronique Générale de Saclay, au commencement de ce travail, et qui a permis la mise en oeuvre du système KOALA, à Monsieur J. WEILL, Chef des Services d'Electronique de Saclay, qui m'a permis de terminer ce travail.

Que Monsieur J. POTTIER, Chef du Service Electronique pour la Recherche et les Applications, trouve ici l'expression de ma profonde gratitude et de mon respect.

Mes remerciements vont aussi à Monsieur AMRAM, Chef de la Section d'Assistance Electronique à la Recherche Fondamentale, et à Monsieur B. de COSNAC, Chef du Groupe d'Initiation et d'Aide à la Conception, dont les critiques et les conseils m'ont été précieux.

Que ceux et celles qui ont permis la publication de ce document (Simone, Tina, l'équipe des Dessinateurs, etc...) trouvent ici l'expression de mes sincères remerciements.

TABLE des MATIERES

			Pages
LIMINAIRE			
INTRODUCTION			1
PREMIERE PARTIE			4
CHAPITRE 1.		GENERALITES	4
	1.1	LES DIFFERENTES FORMES	4
	1.2	LA REDUCTION	6
	1.3	GRAMMAIRE - LANGAGE - ANALYSE SYNTAXIQUE	7
	1.4	PROGRAMME CHOPIN	8
CHAPITRE 2.		PROGRAMME CHOPIN	13
		ANALYSE SYNTAXIQUE DE RECONNAISSANCE	13
	2.1	DEFINITION DE LA GRAMMAIRE	14
		2.11 Le vocabulaire terminal	14
		2.12 Le vocabulaire non terminal (1ère forme)	17
		2.13 Les productions en première forme	21
	2.2	GRAMMAIRE A PRECEDENCE D'OPERATEURS	21
		2.21 Transformation en grammaire d'opérateurs	22
		2.22 Graphes des relations de précédence dans γ_T	22
		2.23 Transformation en grammaire à précédence d'opérateurs	23
		2.24 La liste des productions	24
		2.25 Le tableau des précédences	25
		2.26 Le tableau de poids des éléments terminaux	25
	2.3	LES DEFINITIONS DES GRAMMAIRES PREMIERE ET SECONDE FORMES	26
		2.31 Tableau réduit des précédences	26
		2.32 Tableau réduit des précédences conditionnées	27
		2.33 Signification physique des précédences conditionnées.	28

Pages

CHAPITRE 3. ANALYSE SYNTAXIQUE DE REDUCTION 37

3.1 GENERALITES 37

3.11 Ensemble générateur de $\mathcal{B}$ 38

3.12 Loi de compositions 38

3.13 Racines 39

3.14 Radical 39

3.15 Relation d'ordre 39

3.16 Complémentaires 40

3.17 Consensus 40

3.18 Racine première. Racine et radical de base 40

3.19 Représentation complète. Représentation première 40

3.2 LES DIFFERENTES METHODES DE REDUCTION 41

3.21 Recherche des racines premières 41

3.22 Recherche d'une représentation première 43

3.3 DEVELOPPEMENT DES FORMES BRUTES 44

3.31 Représentation en forme canonique 44

3.32 Représentation en forme complète 45

3.33 Représentation en forme première 46

3.34 Représentation choisie : la forme allégée 46

3.4 ANALYSE SYNTAXIQUE DE REDUCTION 47

3.41 Ensemble des réductions syntaxiques 47

3.42 Réductions des chaînes terminales 48

SECONDE PARTIE - DESCRIPTION DU PROGRAMME CHOPIN 54

CHAPITRE 4. ECONOMIE ET ORGANISATION GENERALE 54

4.1 INITIALISATION 55

4.2 CONSTITUTION DU VOCABULAIRE TERMINAL 56

4.3 ANALYSE DE RECONNAISSANCE 56

4.4 ANALYSE DE REDUCTION 57

4.41 Allégement 57

4.42 Réduction 58

Pages

4.5 IMPRESSION 59
4.51 Impression de la représentation obtenue 59
4.52 Impression du diagnostic d'erreur 59

4.6 POURSUITE DE L'EXECUTION 59

CHAPITRE 5. CONSTITUTION DU VOCABULAIRE TERMINAL 62
5.1 LES DIFFERENTS ELEMENTS DU VOCABULAIRE TERMINAL 63
5.11 Symboles opératoires. Ponctuation 63
5.12 Identificateurs de variables 64
5.13 Présentation de la liste des éléments en entrée 64

5.2 CONSTITUTION DU VOCABULAIRE TERMINAL 65
5.21 Symboles opératoires 65
5.22 Variables 65
5.23 Impression des éléments du vocabulaire terminal 66

5.3 MODIFICATION DU VOCABULAIRE TERMINAL 66

CHAPITRE 6. ANALYSE DE RECONNAISSANCE 68
6.1 GRAMMAIRE "MACHINE" 69
6.11 Vocabulaire non terminal 69
6.12 Ensemble des productions 69
6.13 Table des précédences conditionnées 70
6.14 Vocabulaire terminal 70

6.2 DESCRIPTION DE L'ANALYSE DE RECONNAISSANCE 74
6.21 Eléments terminaux comparables 74
6.22 Recherche aval 74
6.23 Recherche amont 75

6.3 REPRESENTATION MACHINE D'UNE EXPRESSION EN FORME BRUTE 76
6.31 Tableaux de la représentation 77
6.32 Premier membre 77
6.33 Second membre 77
6.34 Exemple de représentation 78

Pages

6.4 ORGANISATION FONCTIONNELLE 78
6.41 Sous-programme TRADUC 78
6.42 Sous-programme MVMNTS 79
6.43 Description fonctionnelle 79

CHAPITRE 7. "REPRESENTATION MACHINE" DU TREILLIS BOOLEEN NORMAL 91
7.1 REPRESENTATION ENSEMBLISTE 91
7.11 Ensembles en correspondance 91
7.12 Homomorphisme défini par $\mathcal{C}$ 92

7.2 REPRESENTATION MACHINE 94
7.21 Relation d'ordre 94
7.22 Racine nulle, racine unité 95
7.23 Composition primaire de deux racines 96
7.24 Composition secondaire de deux racines 97

CHAPITRE 8. ANALYSE DE REDUCTION 99
8.1 ENSEMBLE DES REDUCTIONS 99
8.2 DESCRIPTION DE L'ANALYSE DE REDUCTION 100
8.21 Recherche amont 100
8.22 Recherche aval 102
8.23 Fin de l'analyse syntaxique de réduction 104

8.3 REDUCTION DE LA FORME ALLEGEE 104
8.31 Recherche de la représentation complète 104
8.32 Recherche d'une représentation première 107

8.4 DESCRIPTION FONCTIONNELLE 108
8.41 Recherche de la forme allégée 108
8.42 Recherche des représentations complète et première 110

CONCLUSION 118

BIBLIOGRAPHIE 121

ANNEXE. 123

En forme de ....

L I M I N A I R E

Avant qu'il n'aborde ce document, nous désirons fournir quelques indications au lecteur.

Pourquoi ce nom de CHOPIN ? Lorsque nous abordâmes cette étude, notre ambition se limitait au traitement en première forme, avec l'assistance d'un ordinateur, des expressions booléennes que manipule l'électronicien ou l'électrotechnicien. Très vite, nous nous aperçûmes qu'elle n'était pas réaliste. La technologie actuelle exige l'emploi des première et seconde formes, sans distinction. Très rapidement les parenthèses, c'est-à-dire les groupements de termes, pointèrent l'oreille de leur symbole et des difficultés à les envisager. Une manière élégante de les traiter, de contourner l'obstacle consiste à noter les expressions en forme POLONAISE ....

Nous envisageâmes, alors, le problème dans l'optique des langages. Ce point de vue quelque peu nouveau pour nous a présenté des difficultés, mais nous a procuré aussi des joies et des plaisirs. Plaisir de l'exploration d'un nouveau domaine, de la découverte de nouvelles possibilités, de l'existence de liens entre disciplines apparemment étrangères. Nous avons voulu que ces obstacles, ces plaisirs puérils, ou naïfs quelquefois, transparaissent dans notre compte rendu.

Cependant, pour ne pas être, de trop, fastidieux nous avons séparé la description de l'étude proprement dite, des énoncés et démonstrations sur lesquelles elle s'appuie. Ceux-ci sont groupés en annexe.

Pour ne pas alourdir la présentation, la rédaction et la lecture, nous avons préféré grouper les références bibliographiques, en les situant a posteriori.

L'ensemble du programme CHOPIN est rédigé en langage FORTRAN. Ce fut pour nous une contrainte, semblable au port d'un carcan. Elle était volontaire. L'emploi d'un langage, plus proche de celui de l'ordinateur, présente l'inconvénient majeur de limiter les possibilités de diffusion du programme. L'adaptation d'un programme à un langage conduit, généralement, à réécrire ce programme en entier.

Au cours du texte, les références aux axiomes, théorèmes, corollaires sont faites sous deux formes :

- l'une où le terme est suivi d'une indication chiffrée. La partie gauche caractérise la partie du document où se trouve l'énoncé (numéro du chapitre, ou de l'annexe si cette indication est précédée de la lettre A). La partie droite marque le rang dans la catégorie. Exemples : Théorème 77, Axiome A 81 : septième théorème du chapitre 7, premier axiome de l'annexe 8.

- l'autre sous forme d'indications entre chevrons. La partie gauche, avant le tiret, caractérise l'énoncé A xiome, T héorème, C orollaire. La partie droite en est la numérotation référentielle, établie selon le même principe que ci-dessus. Exemples : <T- 77> , <A- 81> sont les équivalents des deux exemples précédents.

I N T R O D U C T I O N

L'invention du transistor, ses répercussions sur la technologie des composants ont permis, dans le domaine de l'électronique, un accroissement de densité des éléments tant actifs que passifs et, par là même, de la densité des groupements fonctionnels. La complexité, tant de définition que d'étude du plus petit sous-ensemble technologique (une plaquette par exemple), se trouvait prendre des dimensions inaccoutumées (elle passait de l'élément de division par deux à un élément diviseur par 16, ou plus, par exemple).

La maîtrise de la technologie des semi-conducteurs, les impératifs logistiques ont conduit très rapidement aux circuits intégrés logiques et accentué, encore plus, ces deux caractères : densité et complexité. La concurrence commerciale, à laquelle se sont livrés les fabricants de circuits intégrés, ajoutée aux progrès technologiques, a provoqué une floraison de "fonctions" plus élaborées, voire de véritables ensembles de traitement (compteur-décompteur, mémoires mortes...). Le plus petit sous-ensemble technologique voit donc ses dimensions plus fortement accrues.

Les conséquences, dont les répercussions sont sensibles pour le réalisateur, sont multiples :

- complexité des "fonctions" disponibles,
- complexité du plus petit sous-ensemble technologique définissable,
- contrainte temporelle.

Les dimensions des problèmes à traiter atteignent alors des dimensions "sur-humaines".

Fort heureusement, les machines de traitement de l'information subissaient simultanément des améliorations. Elles étaient utilisées tout d'abord pour simuler le fonctionnement de nouvelles machines, puis pour la préparation de leur réalisation. Ainsi, s'amélioraient les conditions d'efficacité, de rentabilité de la conception assistée par ordinateur.

Dans le système KOALA (Conception Assistée par Ordinateur d'ensembles électroniques Logiques), nous envisagions de traiter complètement la réalisation d'un ensemble électronique avec l'aide d'un ordinateur, dans l'optique d'un

utilisateur de circuits intégrés logiques et non d'un fabricant de tels circuits.

Généralement, l'ensemble à envisager ressort du domaine séquentiel qui englobe une partie proprement séquentielle et une partie combinatoire. En envisageant le problème sous cet angle, le traitement d'un problème de logique combinatoire est un cas particulier du traitement d'une logique séquentielle.

L'énoncé du problème doit pouvoir être envisagé sous ces multiples aspects :

- à l'aide de chronogrammes
- à l'aide de matrices de phases
- à l'aide de graphes
- à l'aide d'équations
- à l'aide de schémas.

La première tâche qui sera demandée à l'ordinateur sera de simplifier cet énoncé autant que faire se peut, afin que le réalisateur ait la possibilité de vérifier que l'énoncé est correct ou complet. Puis, l'ordinateur pourra traiter la synthèse de l'ensemble, qui se décompose en :

- une recherche du codage optimum du système séquentiel, donc l'obtention des expressions;
- une synthèse de la logique combinatoire, soit en envisageant des fonctions indépendantes, soit en envisageant des multi-fonctions.

Avant de poursuivre, il est souhaitable de procéder à une vérification. Elle pourra se faire soit par une analyse de l'ensemble, soit par une simulation. Le fonctionnement escompté étant obtenu, l'ordinateur peut procéder alors à l'implantation, puis au tracé automatique du câblage.

Il restera à l'ordinateur deux autres tâches à remplir :

- rechercher les séquences d'essais en vue de la maintenance de l'équipement envisagé;
- éditer les différents documents utiles à la réalisation de celui-ci.

Mais, d'autres tâches annexes peuvent être aussi envisagées et l'une d'entre elles est l'analyse de circuits intégrés logiques existants. Elle peut être envisagée comme un mode de fonctionnement particulier de KOALA : introduction d'expressions logiques sous forme brute pour représenter le circuit, et traitement de ces expressions pour obtenir soit une expression réduite, soit une matrice des phases réduites.

La comparaison d'expressions, en vue d'une recherche d'équivalence, présente de l'intérêt en fin de synthèse séquentielle pour rechercher quelles peuvent être les fonctions séquentielles disponibles les plus appropriées au circuit étudié.

Sous-ensembles du système KOALA, le programme CHOPIN a été étudié pour traiter des expressions sous leur forme brute et donner une expression réduite, ainsi que la liste de tous les éléments premiers, préparant soit l'analyse du circuit, soit la recherche d'équivalences, permettant ainsi une formulation de synthèse.

PREMIERE PARTIE

- CHAPITRE 1 -

G E N E R A L I T E S

Nous allons, à partir de deux exemples, faire apparaître les notions générales indispensables pour l'exposé. Ces notions seront développées, soit dans des chapitres annexes, soit dans le texte principal.

Proposons-nous d'étudier le circuit, représenté par le schéma de la figure 1.1. Dans une première étape, nous établirons les expressions des variables de sortie, éléments de

$$Z=\{c_r, f_0, f_1, f_2, f_3, g, p, plu\}$$

ensemble des "variables de sortie" fonction des éléments de :

$$X=\{a_0, a_1, a_2, a_3, b_0, b_1, b_2, b_3, c_n, m, s_0, s_1, s_2, s_3\}$$

ensemble des variables d'entrée ou variables principales. Nous envisagerons alors de déterminer le fonctionnement du circuit, compte tenu de ces expressions.

1.1 LES DIFFERENTES FORMES

Considérons, uniquement, f_1 . Nous pouvons l'écrire sous une "forme brute" :

$$f_1=(((a_0+b_0.s_0+b_0'.s_1)'+(a_0.b_0.s_3+a_0.b_0'.s_2)').((a_0+b_0.s_0+b_0'.s_1)'.(a_0.b_0.s_3+a_0.b_0'.s_2)')'+(m.c_n)').(((a_0+b_0.s_0+b_0'.s_1)'+(a_0.b_0.s_3+a_0.b_0'.s_2)').((a_0+b_0.s_0+b_0'.s_1)'.(a_0.b_0.s_3+a_0.b_0'.s_2))'.(m.c_n)')' \qquad (1.1)$$

Note :

Le support des données est la carte perforée. Le surlignage s'avère donc impossible.

D'une façon générale, la complémentation est représentée en dernier, lors de l'écriture manuelle d'expressions logiques.

Aussi, avons-nous convenu de placer son symbole représentatif (ici ') après.

Cette représentation littérale est difficilement utilisable; une réduction de l'expression est souhaitable. Encore faut-il connaître la forme de représentation souhaitée :

- réunion d'intersections ou première forme;
- intersection de réunions ou seconde forme ?

Dans l'un ou l'autre cas, la réduction suppose un développement qui tienne compte de la priorité accordée :

- soit à l'intersection, pour la première forme,
- soit à la réunion, pour la seconde forme.

Le traitement de l'expression (1.1) serait ici long et fastidieux. Nous allons envisager une expression plus simple :

$$z = ((m'.(a'_4 + b'_4)')'.(a'_4.b'_4)')' \qquad (1.2)$$

Le traitement de cette expression va être différent selon :

- le type de forme envisagée,
- le sens d'exploration de l'expression (rappelons que manuellement ce traitement se fera d'une manière globale et que, mécaniquement, il se fera pas à pas, c'est-à-dire caractère par caractère).

Les figures, 1.2a et 1.2b, représentent ce traitement en première forme, exploration gauche-droite et droite-gauche respectivement; les figures 1.3 le traitement en seconde forme.

Leur examen nous conduit à deux conclusions immédiates :

- l'exploration droite-gauche est de loin préférable,
- une précédence existe entre les opérateurs (nous verrons plus loin que dans le cas considéré, la grammaire est une grammaire à précédence d'opérateurs): certaines opérations sont effectuées avant d'autres.

Nous pouvons faire d'autres remarques :

- le traitement ne peut fructueusement être envisagé que si l'expression est correctement représentée, c'est-à-dire si ses limites ont été reconnues et si aucune erreur de syntaxe ne s'est présentée.

- ce traitement doit s'accompagner d'un allégement des résultats partiels. Les résultats sont donnés compte tenu de ces allégements.

1.2 LA REDUCTION

Réduire une expression logique consiste à

- rechercher l'ensemble $\mathcal{E}$ des implicants (ou couvertures) premiers
- rechercher une base $\mathcal{B}$ (ou enveloppe) complète.

Dans le cas d'une expression, telle que celle donnée en (1.1), nous pourrions envisager de réduire, systématiquement, toute sous-expression en sa base complète. Les opérations ne s'effectueraient que sur des expressions réduites. Une telle procédure serait, d'une part, longue et coûteuse; d'autre part, elle ne permettrait pas la simple comparaison de deux expressions. Une telle comparaison ne peut se faire qu'au niveau de $\mathcal{E}$.

Cependant, la recherche de $\mathcal{E}$ est facilitée si :

- aucun élément de l'expression n'est redondant,
- il n'existe aucune paire d'éléments liés par une relation d'implication. De tels éléments peuvent exister dans des sous-expressions, donc, a fortiori, dans la forme brute. Leur existence dans des sous-expressions conduit à la génération soit d'éléments redondants, soit de nouveaux couples liés. Il est donc souhaitable d'alléger les expressions au fur et à mesure de leur création pour obtenir une expression terminale allégée qui est plus facilement réductible.

1.3 GRAMMAIRE - LANGAGE - ANALYSE SYNTAXIQUE

Décrire le processus mental qui nous permet de comprendre un texte, écrit ou oral, serait non seulement hors de propos, mais utopique, eu égard à nos connaissances. Cependant, nous pouvons analyser et mettre en évidence les différentes phases du processus, encore que leur importance et leur ordre chronologique varient d'un individu à l'autre. Considérons donc un texte écrit, réduisant ainsi les difficultés et nous plaçant plus près de notre centre d'intérêt.

Nous diviserons le texte en parties. Nous examinerons chacune d'entre elles séparément, dans un certain ordre. Nous recommencerons cette subdivision jusqu'à arriver à l'élément, le plus simple, qui conduise à la compréhension : la proposition. Cette opération de subdivision est facilitée, pour ne pas dire quelquefois imposée, par la présentation typographique, l'organisation en chapitres, paragraphes, alinéas, par les signes de ponctuation et par le rôle privilégié de certains mots (non par leur sens, mais par leur rôle d'éléments syntaxiques).

Pour affirmer le sens d'une proposition nous sommes, quelquefois (toujours lorsque la connaissance de la langue est imparfaite), amenés à en faire l'analyse syntaxique : déterminer les sujet, verbe, complément etc....., avant de leur attribuer leur valeur sémantique. Nous procédons alors :

- à une réécriture mentale de la proposition : nous la réexprimons dans notre propre système d'expression

  - soit en la résumant,
  - soit en la gardant telle quelle,
  - soit en l'éliminant parce qu'elle ne nous semble pas modifier la compréhension des autres propositions.

- à un rejet, si la construction défectueuse conduit à une ambiguïté. Encore que nous reprendrons cette réexpression après l'analyse d'autres propositions. Il y a là une dépendance contextuelle.

De proche en proche, nous remonterons au chapitre puis au texte complet.

Tout ceci suppose la connaissance et du vocabulaire, et de la grammaire.

Si nous considérons maintenant l'étu . d'une expression logique, nous pouvons mettre en évidence une similitude :

- il est inutile d'envisager les opérations si l'expression est incorrectement écrite : mauvaise succession des signes opératoires, variables inconnues etc....

- nous effectuerons certaines opérations, certains groupements, eu égard à une précédence entre opérateurs. Ces groupements sont délimités par certains signes opératoires.

- l'opération s'arrêtera à un certain moment. Celui où nous aurons mis en évidence tous les éléments nécessaires à la caractérisation de la fonction.

Nous pouvons pousser cette similitude jusqu'au bout et essayer de définir un langage, une grammaire et un vocabulaire qui nous faciliteraient l'étude de ces expressions. Ce langage devra nous permettre :

- de reconnaître la construction correcte :
  - succession des signes opératoires,
  - identification des variables,
  - terminaison,

- d'effectuer les opérations, c'est-à-dire de déterminer les phrases premières et l'ordre des opérations.

Ce langage doit être un langage à structure de phrases, à indépendance contextuelle et à précédence d'opérateurs(v. Annexes 1 et 2).

1.4 PROGRAMME CHOPIN

A la lumière de ce qui précède, nous voyons se dessiner

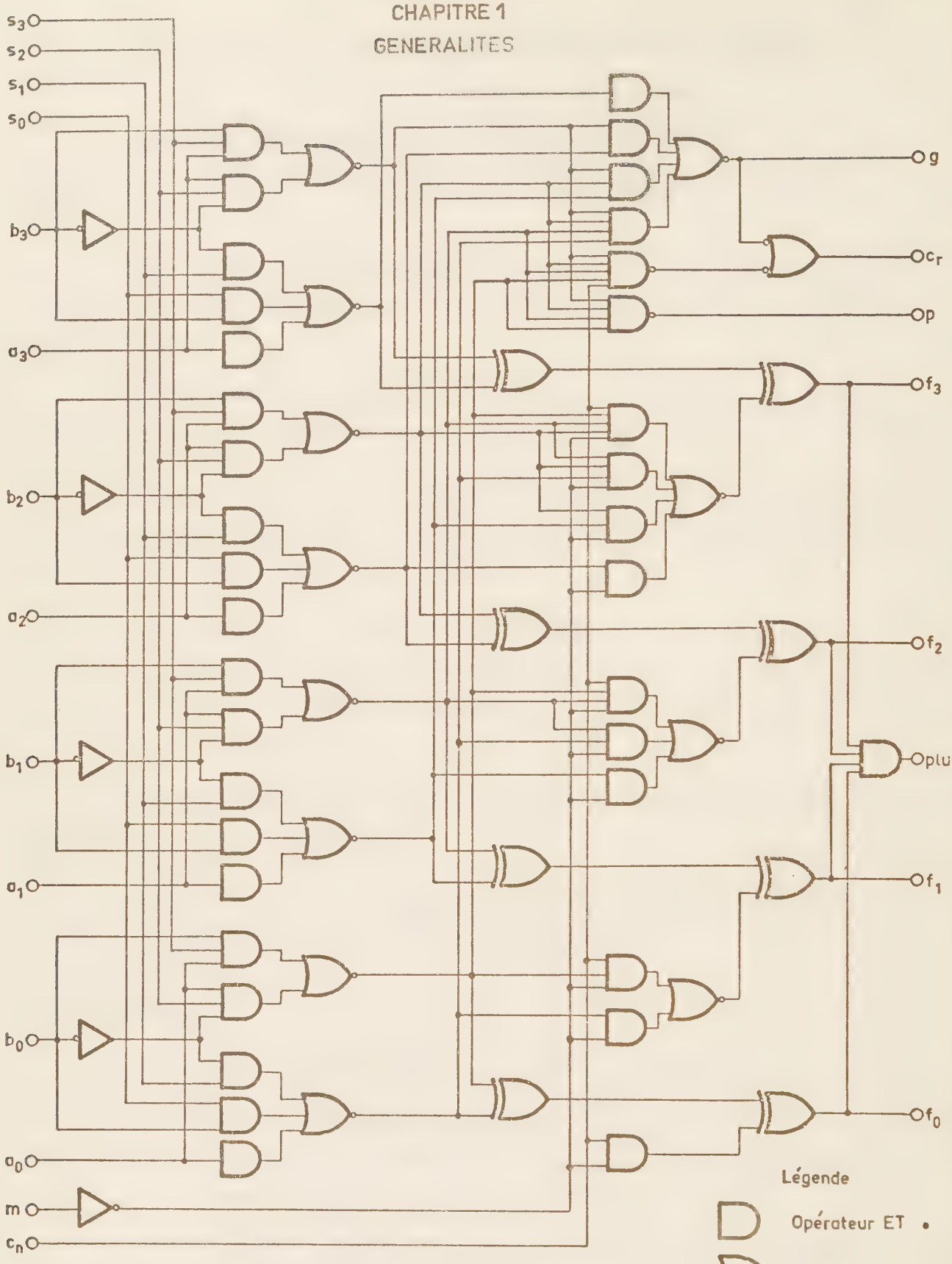
- l'organisation générale du programme CHOPIN :
  - vérification d'écriture correcte, ou analyse syntaxique de reconnaissance;
  - analyse syntaxique de réduction, dont le produit final est l'expression complète de la fonction;

- le plan d'exposition de ce document :

    - étude de la grammaire,

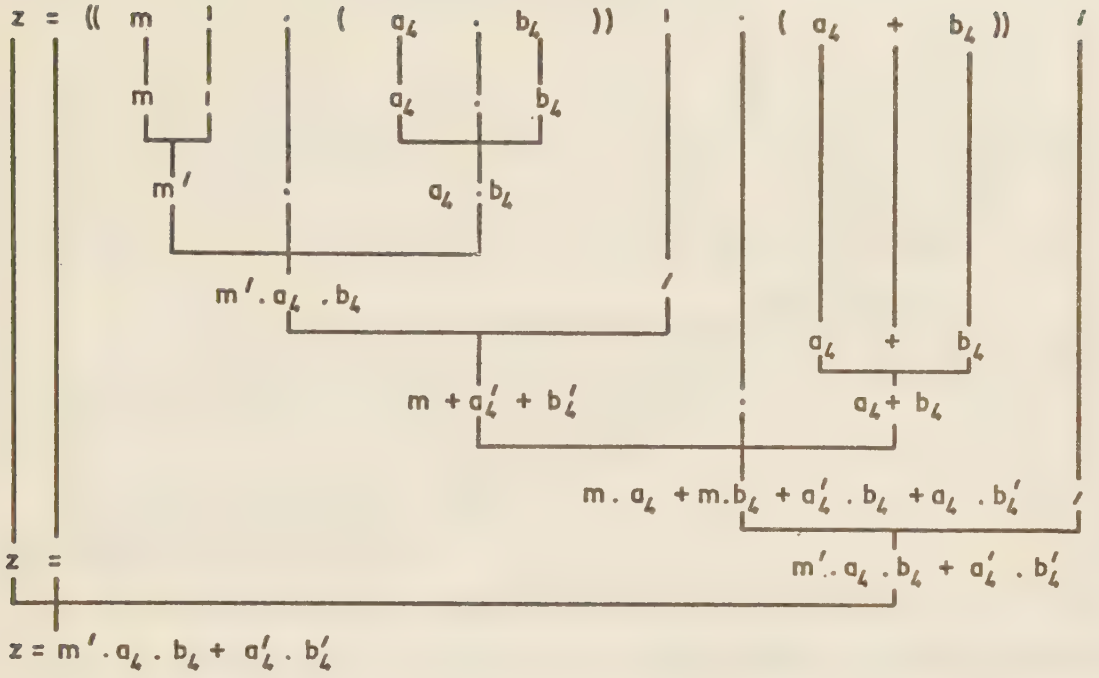
    - étude de l'algorithme d'allégement et de réduction,

    - description du programme CHOPIN.



EXEMPLE DE CIRCUIT A ETUDIER

FIG. 1-1



_a_ Exploration gauche - droite

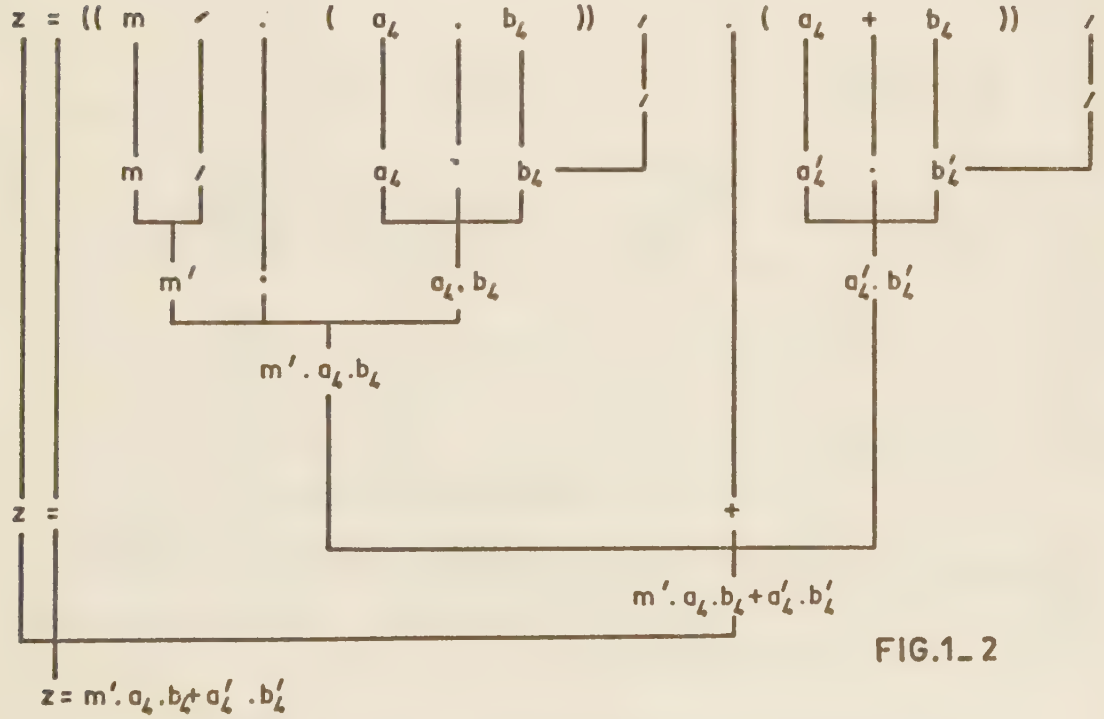
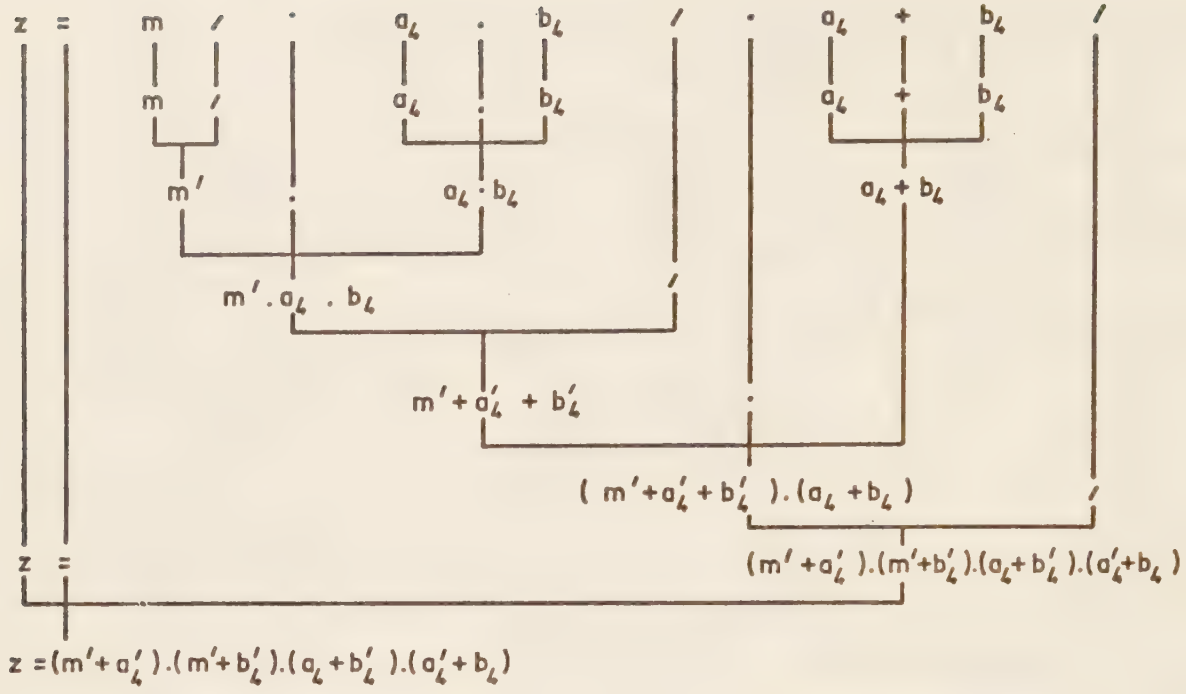


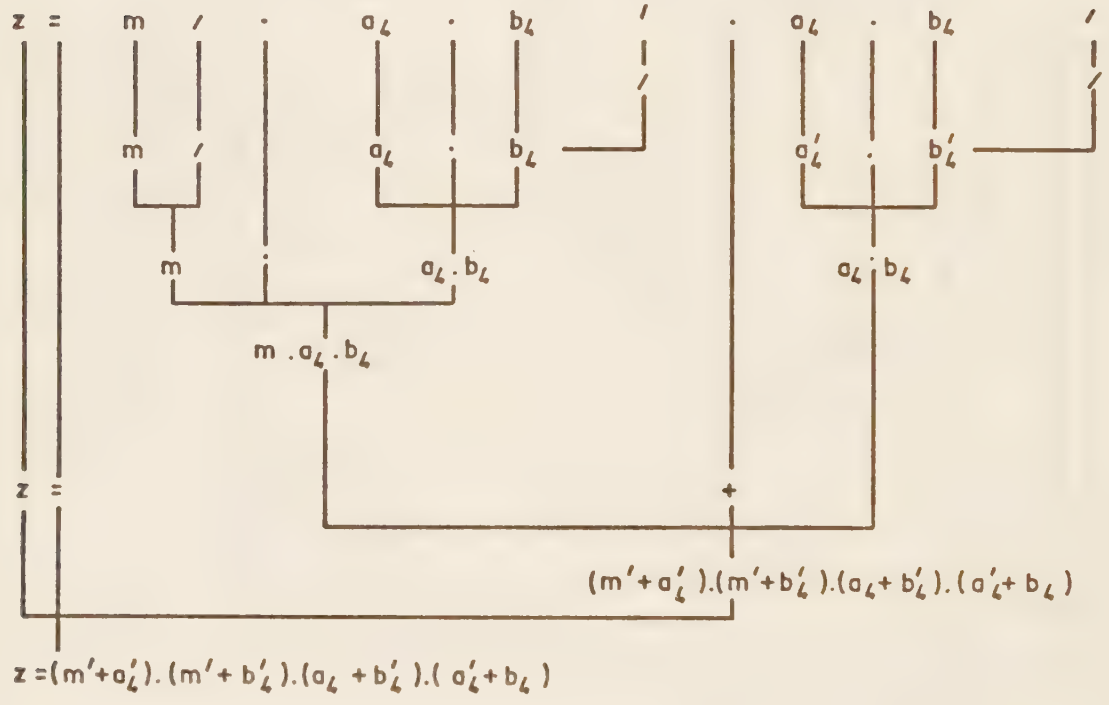
FIG.1_2

_b_Exploration droite-gauche

TRAITEMENT MANUEL D'UNE EXPRESSION LOGIQUE
EN PREMIERE FORME



_ a _ Exploration gauche-droite



_ b _ Exploration droite-gauche

FIG.1_3

TRAITEMENT MANUEL D'UNE EXPRESSION LOGIQUE
EN SECONDE FORME

- CHAPITRE 2 -

PROGRAMME C H O P I N

ANALYSE SYNTAXIQUE DE RECONNAISSANCE

Pour alléger les conditions d'exploitation, faciliter l'utilisation de données existantes, laisser une plus grande liberté de présentation des données, nous avons convenu dans l'écriture des expressions logiques "données en entrée" :

- de rendre non significatifs les blancs ou espacements;
- de ne pas fixer le nombre de caractères (dans les limites imposées toutefois) des identificateurs de chaque variable.

Nous nous sommes, d'autre part, astreints à conserver le mode traditionnel d'écriture en utilisant :

- un premier membre qui définit la "fonction";
- un séparateur de membres que nous représenterons arbitraitement par :=
- un second membre, sans aucun symbole qui en détermine la fin;
- la complémentation des variables, ou groupements de variables, a posteriori (post-fixation).

Ces contraintes (ou conventions) nous ont conduit à :

- délimiter les expressions a posteriori;
- n'envisager les réductions, qu'une fois cette délimitation obtenue et la vérification de présentation correcte effectuée.

Deux analyses sont envisagées pour chaque expression :

- une analyse ascendante, gauche-droite, au cours de laquelle les délimiteurs sont introduits, le codage machine et la recherche des erreurs syntaxiques effectués : l'analyse de reconnaissance.

- une seconde analyse ascendante, droite-gauche avec réduction (au sens grammatical du terme) : l'analyse de réduction.

2.1 DEFINITION DE LA GRAMMAIRE

Il est possible de transformer une grammaire en grammaire à précédence d'opérateurs et, ainsi, faciliter la recherche des structures.

Caractériser une grammaire impose la définition :

- d'un vocabulaire terminal : V_T ;
- d'un vocabulaire non terminal : V_N ;
- d'un ensemble de productions : P ;
- d'un axiome : S .

L'utilisation d'une grammaire à précédence d'opérateurs demande, en plus de la définition précédente, la connaissance des relations $<\cdot$, $\doteq$, $\cdot>$ entre éléments terminaux (Annexe 3).

2.11 Le vocabulaire terminal

V_T est l'ensemble des symboles utilisés pour écrire les éléments du langage défini sur la grammaire.

Nous distinguerons trois parties de l'ensemble V_T :

- V_{TS} : ensemble de tous les symboles employés pour écrire les identificateurs, représentant les variables;
- V_{TO} : ensemble de tous les symboles qui représentent les opérations logiques booléennes (réunion, intersection, complémentation ....);
- V_{TP} : ensemble de tous les symboles représentatifs des "signes de ponctuation".

Ces trois sous-ensembles constituent une partition de V_T.

2.111 Les identificateurs de variables

Les identificateurs de variables doivent, quelquefois, permettre de reconnaître une caractéristique de celles-ci. Un sigle est alors utile.

Nous avons, donc, pris pour éléments de $\mathcal{V}_{TS}$ tous les caractères alpha-numériques :

$$\mathcal{V}_{TS} = \{ \text{caractères alpha-numériques} \}$$

Nous verrons que si pour la "déclaration" des identificateurs le blanc ou espacement n'est pas élément de $\mathcal{V}_{TS}$, il n'en est pas de même au niveau de l'écriture des expressions (cf. infra).

2.112 Les symboles opératoires

Les symboles opératoires ne sont pas imposés. L'utilisateur du programme les définira par une déclaration, avec cependant deux contraintes :

- ils ne doivent faire intervenir qu'un caractère;
- ce caractère ne doit pas être un élément de $\mathcal{V}_{TS}$.

Note

Pour tout ce qui concerne ce document, nous prendrons :

$$\mathcal{V}_{TO} = \{ \cdot , + , ' \}$$

avec	.	ET	opération d'intersection
	+	OU	opération de réunion
	'	PAS	opération de complémentation.

2.113 Les signes de ponctuation

Nous appelons, ainsi, les symboles nécessaires à l'écriture des expressions.

Comme pour les symboles opératoires, à l'exception toutefois des délimiteurs :

- ils ne doivent avoir qu'un caractère;
- ce caractère ne doit pas être élément de $\mathcal{V}_{TS}$ ou de $\mathcal{V}_{TO}$.

2.113.1 Les séparateurs de membres

Les expressions à traiter sont des égalités, en fait des affectations. Elles peuvent correspondre à une fonction normale ou complémentée, mais aussi à la borne supérieure, ou inférieure, d'une fonction. Nous avons donc prévu trois séparateurs de membres :

<égalité>	ou	<EGAL>
<limite supérieure>	ou	<SUPR>
<limite inférieure>	ou	<INFR>

2.113.2 Les symboles de groupement

Ce sont en fait les parenthèses d'association :

<parenthèse gauche>	ou	<PARG>
<parenthèse droite>	ou	<PARD>

2.113.3 Les délimiteurs

Ce sont les deux symboles de début : <DEBUT> et fin d'expression : <FIN> . Ils sont obtenus implicitement.

Note

Pour ce qui concerne ce document :

- les trois séparateurs de membres, qui ont un rôle identique au niveau de la grammaire, seront symbolisés par :

:= <symbole d'affectation>

- nous prendrons :

$$\mathcal{V}_{TP} = \{:=, (,), \dashv, \vdash\}$$

c'est-à-dire respectivement <symbole d'affectation> , <PARG> , <PARD> , <DEBUT> , <FIN> .

2.12 Le vocabulaire non terminal (première forme)

Nous allons définir simultanément productions et vocabulaire non terminal, en suivant l'ordre de notre étude préalable : selon une analyse ascendante, et en partant des éléments terminaux pour arriver à une expression en première forme.

Le résultat en est une grammaire à structure phrasée.

Nous indiquerons, alors, quelles modifications nous y avons apportées pour la transformer en grammaire à précédence d'opérateurs.

Tous ces résultats ne sont valables que pour la première forme. Leur adaptation à la seconde forme se fait en utilisant le principe de dualité (cf. Annexes 6 et 7). Nous définirons ensuite V_N et P pour la seconde forme, par leur seul énoncé.

2.121 Recherche des productions et éléments syntaxiques

Le blanc, ou espacement, n'est pas considéré comme significatif pour la grammaire proprement dite. Il joue le rôle d'élément neutre pour l'opération de concaténation définie sur V^*. Nous ne le représenterons pas dans la suite du texte, étant entendu qu'il peut intervenir dans n'importe quelle production.

2.121.1 Identificateurs de variables

Les variables sont identifiées à l'aide d'éléments de V_{TS}, avec un degré d'occurence maximum de 4. Mais, nous devons différencier ces variables selon leur rôle dans le circuit (cf. Annexe 5) en :

- variables d'entrée,
- variables secondaires,
- et variables de sortie.

Nous supposerons que cette différentiation est faite en particularisant une lettre de l'alphabet. Nous verrons, plus tard, comment elle est effectivement obtenue.

Symbolisons par m un élément quelconque de V_{TS} :

$m \in \{\text{lettres de l'alphabet latin}\} \cup \{0,1,2,3,4,5,6,7,8,9\}$

Nous définirons ensuite quatre productions :

< caractère >	: : = m	(Pr.1)
< VENT >	: : = x $\{ <\text{caractère}> \}^3$	(Pr.2)
< VSEC >	: : = y $\{ <\text{caractère}> \}^3$	(Pr.3)
< VSOR >	: : = z $\{ <\text{caractère}> \}^3$	(Pr.4)

2.121.2 Premier membre

Les conventions d'écriture, que nous nous sommes imposé (cf.supra), obligent à envisager un premier membre séparé d'un second membre par le symbole :=. Dans ce premier membre, doit se trouver la variable qui est exprimée en fonction des variables du second membre. Cette "variable fonction" ne peut être que :

- soit une variable secondaire,
- soit une variable de sortie,

précédée du symbole ⊣ .

< fonction >	: : = ⊣ < VSEC >	(Pr. 5a)
< fonction >	: : = ⊣ < VSOR >	(Pr. 5b)

mais cette "variable fonction" peut être complémentée - nous ne limiterons pas le nombre de complémentations - :

< fonction >	: : = < fonction >′	(Pr. 5c)

Note :

Les blancs ou espacements n'étant pas significatifs, nous les omettrons systématiquement.

Les productions 5a, 5b, 5c ont pour premier membre le même élément non terminal. Nous écrirons :

< fonction > : : = ⊣ < VSEC > / ⊣ < VSOR > / < fonction >′ (Pr. 5)

Le premier membre sera délimité quand le symbole := sera lu :

< premier membre > : : = < fonction > := (Pr. 6)

2.121.3 Primaire

Dans le traitement en première forme, les opérations d'intersection sont les premières effectuées. Elles font intervenir, ce que nous appellerons, des primaires.

Une primaire peut être :

- une variable, hormis les variables de sortie qui ne sont pas déclarées variables secondaires :

<primaire> : : = <VENT> / <VSEC> (Pr. 7a)

- une réunion de variables ou d'intersections de variables, écrite entre parenthèses :

<primaire> : : = (<réunion>) (Pr. 7b)

- un quelconque de ces éléments complémentés.

Envisageons la production :

<primaire> : : = <primaire>′ (Pr. 7c)

le théorème A22 nous permet d'écrire :

(Pr. 7a) et (Pr. 7c) ⟶ <primaire> : : = <VENT> / <VSEC> (Pr. 7d)
(Pr. 7b) et (Pr. 7c) ⟶ <primaire> : : = (<réunion>) (Pr. 7e)

le théorème A21 est alors satisfait. La production (Pr. 7c) couvre les deux dernières. Nous la conserverons donc, et écrirons pour résumer en une production unique :

<primaire> : : = <VENT> / <VSEC> / (<réunion>) / <primaire>′ (Pr. 7)

2.121.4 Terme

Dans le cas d'expression en première forme, la réunion porte soit sur des primaires, soit sur des intersections de primaires. Nous appellerons termes, ces composants de la réunion.

Un terme peut être :

- un primaire seul,
- l'intersection de deux primaires, d'un primaire et d'un terme, d'un terme et d'un primaire.

Pour les raisons invoquées ci-dessus, nous écrirons :

<terme> : : = <primaire> / <terme> . <primaire> / <primaire>.
<terme> (Pr. 8a)

Mais, l'intersection est commutative. Nous pouvons, donc, supprimer une des deux dernières productions du second membre de (Pr. 8a) et écrire :

<terme> ::= <primaire> /<terme> . <primaire> (Pr.8)

2.121.5 Réunion

L'opération de réunion porte :

- soit sur des primaires,
- soit sur des termes,
- soit sur la réunion de ces mêmes éléments.

Compte tenu de la commutativité de la réunion, nous serions conduits à écrire la liste des productions :

<réunion> : : = <primaire> + <primaire> / <primaire> + <terme> / <primaire> + <réunion> (Pr. 9a)

: : = <terme> + <terme> / <terme> + <réunion> (Pr. 9b)

: : = <réunion> + <réunion> (Pr. 9c)

Nous ne conserverons pas la dernière. Elle risque de produire un cyclage d'opérations. Nous écrirons :

<réunion> : : = <terme> / <réunion> + <terme> (Pr. 9)

qui, eu égard à <T-A22> ,<T-A21> et <C-A21> , couvre les trois groupes de productions précédentes.

2.121.6 Second membre

Le second membre est délimité par le symbole de fin : ⊢ . Il est constitué par l'un quelconque des éléments syntaxiques :

<primaire>
<terme>
ou <réunion> .

Pour les raisons invoquées précédemment, nous représenterons la production qui lui correspond par :

<second membre> : : = <réunion> (Pr. 10)

2.121.7 Expression en première forme

Une expression en première forme est définie par un premier membre et un second membre, donc :

<expression> : : = <premier membre> <second membre> (Pr. 11).

2.122 Le vocabulaire non terminal

Il se déduit du paragraphe précédent :

V_N = { <caractère> , <VENT> , <VSEC> , <VSOR> , <fonction> , <premier membre> , <primaire> , <terme> , <réunion> , <second membre> , <expression> }.

2.13 Les productions en première forme

La liste qui suit résume les différentes productions établies ci-dessus :

m ∈ {lettres de l'alphabet latin} ∪ {0,1,2,3,4,5,6,7,8,9}

<caractère> : : = m (Pr. 1)

<VENT> : : = x { caractère }^{3} (Pr. 2)

<VSEC> : : = y { caractère }^{3} (Pr. 3)

<VSOR> : : = z { caractère }^{3} (Pr. 4)

<fonction> : : = ⊣ <VSEC> / ⊣ <VSOR> / <fonction>′ (Pr. 5)

<premier membre> : : = <fonction> := (Pr. 6)

<primaire> : : = <VENT> / <VSEC> / (<réunion>)/ <primaire>′ (Pr. 7)

<terme> : : = <primaire> / <terme> . <primaire> (Pr. 8)

<réunion> : : = <terme> / <réunion> + <terme> (Pr. 9)

<second membre> : : = <réunion> (Pr. 10)

<expression> : : = <premier membre> <second membre> (Pr. 11)

2.2 GRAMMAIRE A PRECEDENCE D'OPERATEURS

Une première remarque qui s'impose est que la grammaire, définie en prenant <expression> pour axiome, est certes une grammaire à structure phrasée, mais absolument pas une grammaire d'opérateurs :

(Pr.2), (Pr.3), (Pr.4), (Pr.11), <A-A25> : Absurde

chacune de ces productions fait intervenir comme second élément au moins une paire d'éléments non terminaux.

2.21 Transformation en grammaire d opérateurs

La première transformation portera sur les quatre premières productions en supprimant le non-terminal, < caractère> , et en écrivant :

<VENT> : : = x $\{ m \}^5$ (Pr. 2 - 0)

<VSEC> : : = y $\{ m \}^5$ (Pr. 3 - 0)

<VSOR> : : = z $\{ m \}^5$ (Pr. 4 - 0)

La seconde transformera (Pr. 11) pour faire intervenir un élément terminal entre les deux éléments non terminaux : < premier membre> et <second membre> . Le terminal le plus judicieux est := . Ce qui nous conduit à supprimer et (Pr. 6), et l'élément non terminal < premier membre> . Nous écrirons donc :

<expression> : : = <fonction> := < second membre> (Pr. 11 - 0)

2.22 Graphes des relations de précédence dans V_T

Nous pouvons remplacer (cf. A2.24) chaque relation de précédence, dans V_T , autre que $\doteq$, par une composition des relations :

- $\mathcal{R}_{TD}$ (... a, comme élément terminal le plus à droite d'une dérivée directe,...) de V_N vers V_T ;
- $\mathcal{R}_{TG}$ (... a, comme élément terminal le plus à gauche d'une dérivée directe,...) de V_N vers V_T ;
- $\mathcal{R}_{NG}$ (... a, comme élément extremum gauche d'une dérivée directe,...) dans V_N ;
- $\mathcal{R}_{ND}$ (... a, comme élément extremum droite d'une dérivée directe,...) dans V_N ;
- $\mathcal{R}_{AD}$ (... est, dans une dérivée directe d'un élément non terminal, adjacent à droite, de ...) de V_T vers V_N ;
- $\mathcal{R}_{AG}$ (... est, dans une dérivée directe d'un élément non terminal, adjacent à gauche de ...) de V_T vers V_N ;

$$< = \mathcal{R}_{AG} \circ \mathcal{R}^{*}_{NG} \circ \mathcal{R}_{TG}$$

$$> = \mathcal{R}^{*}_{TD} \circ \mathcal{R}^{*+}_{ND} \circ \mathcal{R}^{*}_{AD}$$

ou compte tenu des tableaux booléens associés (cf. Annexe 9) :

$$T_{<\cdot} = T_{A<} \wedge T^{*}_{N<} \wedge T_{T<}$$
$$T_{>} = T^{*}_{TD} \wedge T^{*\alpha}_{ND} \wedge T^{\alpha}_{AD}$$

Les figures 2.1 et 2.2 reproduisent les tableaux. Les ensembles V_T et V_N sont rangés comme suit

$$V_T = \{\dashv, m, y, z, :=, \alpha, (\ ,\ ), +, \cdot, ', \vdash\}$$
$$V_N = \{Pr.2\text{-}0, Pr.3\text{-}0, Pr.4\text{-}0, \ldots\ldots\ldots, Pr.10\text{-}0, Pr.11\text{-}0\}$$

La figure 2.3 représente les diagrammes cartésiens des relations $<\cdot$ et $>$, dans V_T .

Le graphe de la troisième relation $\doteq$, dans V_T , se déduit immédiatement des productions (Pr.2-0), (Pr.3-0), (Pr.4-0) et (Pr. 7) :

$$G_{\doteq} = \{(y,m), (x,m), (z,m), (m,m), (\ (\ ,\ )\ )\}$$

et son diagramme cartésien est reproduit figure 2.3b.

2.23 Transformation en grammaire à précédence d'opérateurs

L'analyse des diagrammes représentés figure 2.3 montre que la grammaire définie ci-dessus n'est pas une grammaire à précédence d'opérateurs :

- $G_{\doteq} \cap G_{>} = \{(m,m)\}$
- $\exists a \in V_T \ ; \dashv > a \ : \dashv >'$ (figure 2.2a)
- $\exists a \in V_T \ ; a <\cdot \vdash \ : :=<\cdot \vdash$ (figure 2.2a)

nous devrons donc modifier les productions qui sont à l'origine de ces relations, c'est-à-dire (Pr.2-0), (Pr.3-0), (Pr.4-0), (Pr.5) et (Pr.10).

2.231 Modification du vocabulaire terminal

Les identificateurs des variables sont spécifiques de chaque cas à étudier, sauf dans les exemples académiques. Cette simple constatation nous permet de modifier les trois premières productions.

L'utilisateur définira en données l'ensemble des variables, en indiquant la liste des variables d'entrée, variables secondaires et variables de sortie. Une même variable peut être variable de sortie et variable secondaire. Nous poserons :

$$\lambda_1 \in \text{VENT} \ , \ \lambda_2 \in \text{VSEC} \quad \text{et} \quad \lambda_3 \in \text{VSOR}$$

et de ce fait :

$$V_{TS} = \text{VENT} \cup \text{VSEC} \cup \text{VSOR}$$

2.232 Modification de l'ensemble $\mathbb{P}$.

Cette définition nouvelle de V_{TS} conduit à écrire

<VENT>	: : = λ_1	(Pr. 2-1)
<VSEC>	: : = λ_2	(Pr. 3-1)
<VSOR>	: : = λ_3	(Pr. 4-1)

La modification de la production (Pr.5) doit intervenir au niveau du terminal.

Nous écrirons :

<fonction> : : = <VSEC> / <VSOR> / <fonction>′ (Pr. 5-1)

Compte tenu de l'axiome A22, et dans un but de simplification, nous supprimerons les productions (Pr.2-1) à (Pr.4-1). Ce qui entraîne :

<fonction>	: : = λ_2 / λ_3 / <fonction>′	(Pr. 5-0)
<primaire>	: : = λ_1/ λ_2 / (<réunion>) / <primaire>′	(Pr. 7-0)

La modification de (Pr.10) consisterait à supprimer le terminal ⊢ . Nous avons préféré supprimer (Pr.6) et remplacer

(Pr. 10) par : <expression> : : = <fonction> :≔ <réunion> (Pr. 10-0)
(Pr. 11) par : <première forme> : : = ⊣ <expression> ⊢ (Pr. 11-00)

2.24 La liste des productions

Nous avons finalement :

<fonction>	: : = λ_2 / λ_3 / <fonction>′	(Pr. 5-0)
<primaire>	: : = λ_1/ λ_2 /(<réunion>) / <primaire>′	(Pr. 7-0)
<terme>	: : = <primaire> / <terme> . <primaire>	(Pr. 8)
<réunion>	: : = <terme> / <réunion> + <terme>	(Pr. 9)
<expression>	: : = <fonction> :≔ <réunion>	(Pr. 10-0)
<première forme>	: : = ⊣ <expression> ⊢	(Pr. 11-00)

2.25 Le tableau des précédences

Les figures, 2.4a et 2.4b, représentent les tableaux booléens composants pour la recherche des tableaux $T_{<\cdot}$ et $T_{\cdot>}$. La figure 2.5 reproduit les diagrammes cartésiens superposés des trois relations dans V_T.

Ces diagrammes constituent, ce qu'on appelle, la "matrice des précédences" ou "tableau des précédences" de la grammaire à précédence d'opérateurs.

2.26 Le tableau de poids des éléments terminaux

Un tel tableau présente l'inconvénient d'avoir un encombrement qui croît comme le carré du cardinal n de V_T. Il est possible dans certains cas de remplacer le tableau des précédences, tableau à n lignes et n colonnes, par un tableau à n lignes et 2 colonnes. Ce tableau permet d'associer à chaque élément terminal x un poids $P_{G(x)}$, si l'élément est considéré comme élément gauche dans une chaîne, et un poids $P_{D(x)}$, s'il est considéré comme élément droit. On a alors :

$$\forall x_i, x_j \in V_T$$

$$x_i <\cdot x_j \rightleftharpoons P_{G(x_i)} < P_{D(x_j)}$$
$$x_i \cdot> x_j \rightleftharpoons P_{G(x_i)} > P_{D(x_j)}$$
$$x_i \doteq x_j \rightleftharpoons P_{G(x_i)} = P_{D(x_j)}$$

Nous avons reproduit figure 2.6 l'algorithme de calcul des valeurs du tableau. Le calcul cesse :

- soit après convergence,
- soit lorsqu'une valeur, double de n, est trouvée pour un poids.

La figure 2.7 reproduit :

- les différentes valeurs prises par les éléments des tableaux au cours de la recherche (a);
- le tableau d'affectation en première et seconde forme (b).

2.3 LES DEFINITIONS DES GRAMMAIRES PREMIERE ET SECONDE FORMES

Nous avons résumé, par la figure 2.8, les éléments de définition des deux grammaires utilisées pour le programme CHOPIN. La comparaison des deux tableaux de précédences et, encore plus, celle des deux tableaux d'affectation des poids (figure 2.7b) fait apparaître la dualité des opérations de réunion et d'intersection. Pour passer d'une forme à l'autre il suffit d'inverser :

- d'une part les lignes + et ., d'autre part les colonnes + et .;
- les seules lignes + et., pour les tableaux d'affectation.

La figure 2.9 reproduit les différents tableaux booléens qui ont servi pour la construction de ces tableaux.

2.31 Tableau réduit des précédences

Quel que soit le moyen envisagé pour déterminer la relation qui existe entre deux éléments terminaux, il présuppose la facile reconnaissance des différents éléments terminaux. Si les symboles opératoires sont en nombre restreint, il n'en est pas de même des identificateurs : il est donc naturel de procéder d'abord à la reconnaissance des symboles opératoires et délimiter, ainsi, les identificateurs. Les recherches de précédence n'ont plus aucune raison de porter sur l'ensemble de $\mathcal{V}_T$. Encore faut-il les transformer pour faire apparaître l'insertion éventuelle des identificateurs, nous reviendrons plus loin sur ce point. Retenons simplement, pour l'instant, la suggestion de réduire les relations de précédence aux seuls éléments de $\mathcal{V}_{T0} \cup \mathcal{V}_{TP}$.

Les tableaux de précédences et tableaux d'affectation des poids sont réduits. Mais demeure le problème de la détection des erreurs de syntaxe (cf. Annexe 4). Réduits ou non, la prééminence en ce domaine reste aux tableaux de précédence, indéniablement. Les tableaux d'affectation de poids ne sont utilisables que pour un langage, correctement exprimé, et pour la réduction des chaînes. Pour reconnaître ces chaînes, il faut pouvoir envisager tous les cas de figures. CHOPIN doit tout autant reconnaître que réduire. Nous avons donc préféré utiliser les tableaux réduits de précédences.

2.32 Tableau réduit des précédences conditionnées

L'intersection (figure 2.10)

- du tableau booléen : $T_{\mathcal{R}}$, associé à la relation : "... est lié, par une des trois relations de précédence, à ...", dans $\mathcal{V}_T$,

- avec le tableau booléen : T_{AT}, associé à la relation : "... peut être adjacent à ...", dans $\mathcal{V}_T$,

fait apparaître une classe d'éléments terminaux. Ceux qui sont en relation de précédence et dont l'adjacence est permise.

Nous pouvons envisager, pour les autres éléments terminaux en relation de précédence, les cas suivants :

- " et sont séparés par un élément terminal λ_1 " (relation $\mathcal{R}_{\lambda_1}$, dans $\mathcal{V}_T$)
- " et sont séparés par un élément terminal λ_2 " (relation $\mathcal{R}_{\lambda_2}$, dans $\mathcal{V}_T$)
- " et sont séparés par un élément terminal λ_3 " (relation $\mathcal{R}_{\lambda_3}$, dans $\mathcal{V}_T$).

L'étude des tableaux booléens associés (voir figure 2.11) confirme les résultats avancés dans l'Annexe 3 : il existe deux grandes catégories d'éléments terminaux délimiteurs de propositions. L'une ne peut intervenir que pour des propositions dont l'élément terminal est de type λ_1 ou λ_2, l'autre pour des propositions dont l'élément terminal est de type λ_2 ou λ_3. C'est-à-dire à une délimitation, de facto, des premier et second membres de l'expression.

Enfin, les éléments terminaux $\dashv$ et $\vdash$ sont introduits implicitement. Aucune erreur ne peut intervenir du fait de leur insertion.

Nous avons, donc, été conduits à envisager les sept relations de précédence conditionnée suivantes :

$$\lessdot \;:\; \lessdot \cap (\mathcal{R}_{\lambda_2} \cup \mathcal{R}_{\lambda_3})$$
$$\lessdot \;:\; \lessdot \cap (\mathcal{R}_{\lambda_1} \cup \mathcal{R}_{\lambda_2})$$
$$\gtrdot \;:\; \gtrdot \cap (\mathcal{R}_{\lambda_1} \cup \mathcal{R}_{\lambda_2})$$
$$\doteq \;:\; \doteq \cap (\mathcal{R}_{\lambda_1} \cup \mathcal{R}_{\lambda_2})$$
$$\lessdot \;:\; \lessdot \cap \mathcal{R}_{AT}$$
$$\gtrdot \;:\; \gtrdot \cap \mathcal{R}_{AT}$$

qui résultent des tableaux représentés figures 2.10 et 2.11.

2.33 Signification physique des précédences conditionnées

A celle des relations de précédence (cf. Annexe 3) s'ajoute plus particulièrement chacune des propriétés des relations $\mathcal{R}_{\lambda_1}, \mathcal{R}_{\lambda_2}$ et $\mathcal{R}_{\lambda_3}$. Elles imposent de reconnaître l'adjacence des éléments de $V_{TS} \cup V_{TP}$ ou de leur séparation par l'identificateur d'une variable d'entrée (λ_1), d'une variable secondaire (λ_2) ou d'une variable (fonction) de sortie (λ_3).

Nous couvrons, ainsi, la reconnaissance de toutes les erreurs de syntaxe ou d'analyse, autres que les erreurs de programmation.

GRAMMAIRE D'OPERATEURS

	2	3	4	5	7	8	9	10	11
⊥	0	1	1	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0
y	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
:=	0	0	0	0	0	0	0	1	0
x	0	0	0	0	0	0	0	0	0
(	0	0	0	0	0	0	1	0	0
)	0	0	0	0	0	0	0	0	0
+	0	0	0	0	0	1	0	0	0
•	0	0	0	0	1	0	0	0	0
,	0	0	0	0	0	0	0	0	0
⊢	0	0	0	0	0	0	0	0	0

T_{AG}

	2	3	4	5	7	8	9	10	11
2	1	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0
7	1	1	0	0	1	0	0	0	0
8	1	1	0	0	1	1	0	0	0
9	1	1	0	0	1	1	1	0	0
10	1	1	0	0	1	1	1	1	0
11	0	0	0	1	0	0	0	0	1

T^{*}_{NG}

	⊥	m	y	3	:=	x	(	)	+	•	,	⊢
2	0	0	0	0	0	1	0	0	0	0	0	0
3	0	0	1	0	0	0	0	0	0	0	0	0
4	0	0	0	1	0	0	0	0	0	0	0	0
5	1	0	0	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	1	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	0	1	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	1
11	0	0	0	0	1	0	0	0	0	0	0	0

T_{TG}

TABLEAUX BOOLEENS COMPOSANTS DU TABLEAU $T_{\lessdot}$ ASSOCIE A LA RELATION $\lessdot$

— FIG_2.1a —

	2	3	4	5	7	8	9	10	11
⊥	0	0	0	1	0	0	0	0	0
m	1	1	1	0	0	0	0	0	0
y	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0
:=	0	0	0	0	0	0	0	0	1
x	0	0	0	0	0	0	0	0	0
(	0	0	0	0	0	0	0	0	0
)	0	0	0	0	1	0	0	0	0
+	0	0	0	0	0	0	1	0	0
•	0	0	0	0	0	1	0	0	0
,	0	0	0	1	1	0	0	0	0
⊢	0	0	0	0	0	0	0	1	0

T^{*}_{TD}

	2	3	4	5	7	8	9	10	11
2	1	0	0	0	1	1	1	0	0
3	0	1	0	1	1	1	1	0	0
4	0	0	1	1	0	0	0	0	0
5	0	0	0	1	0	0	0	0	0
7	0	0	0	0	1	1	1	0	0
8	0	0	0	0	0	1	1	0	0
9	0	0	0	0	0	0	1	0	0
10	0	0	0	0	0	0	0	1	1
11	0	0	0	0	0	0	0	0	1

T^{**}_{ND}

	⊥	m	y	3	:=	x	(	)	+	•	,	⊢
2	0	1	0	0	0	0	0	0	0	0	0	0
3	0	1	0	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0	0	0	1	0
7	0	0	0	0	0	0	0	0	0	0	1	0
8	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	0	0	0	0	0	1	1	0	0	1
10	0	0	0	0	1	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0

T^{*}_{AD}

TABLEAUX BOOLEENS COMPOSANTS DU TABLEAU $T_{\gtrdot}$ ASSOCIE A LA RELATION $\gtrdot$

FIG_2.1b

CHAPITRE 2

ANALYSE SYNTAXIQUE DE RECONNAISSANCE

GRAMMAIRE D'OPERATEURS

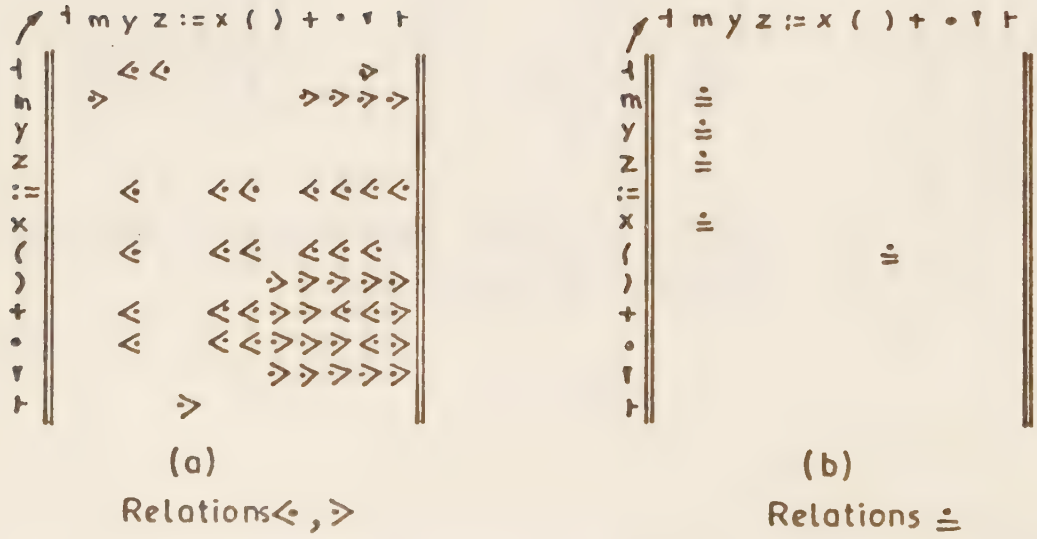
	⊣	m	y	z	:=	x	(	)	+	•	▼	⊢
⊣	0	0	1	1	0	0	0	0	0	0	0	0
m	0	0	0	0	0	0	0	0	0	0	0	0
y	0	0	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0
:=	0	0	1	0	0	1	1	0	1	1	1	1
x	0	0	0	0	0	0	0	0	0	0	0	0
(	0	0	1	0	0	1	1	0	1	1	1	0
)	0	0	0	0	0	0	0	0	0	0	0	0
+	0	0	1	0	0	1	1	0	0	1	1	0
•	0	0	1	0	0	1	1	0	0	0	1	0
▼	0	0	0	0	0	0	0	0	0	0	0	0
⊢	0	0	0	0	0	0	0	0	0	0	0	0

$T_{\lessdot\bullet}$

	⊣	m	y	z	:=	x	(	)	+	•	▼	⊢
⊣	0	0	0	0	0	0	0	0	0	0	1	0
m	0	1	0	0	0	0	0	1	1	1	1	1
y	0	0	0	0	0	0	0	0	0	0	0	0
z	0	0	0	0	0	0	0	0	0	0	0	0
:=	0	0	0	0	0	0	0	0	0	0	0	0
x	0	0	0	0	0	0	0	0	0	0	0	0
(	0	0	0	0	0	0	0	0	0	0	0	0
)	0	0	0	0	0	0	0	1	1	1	1	1
+	0	0	0	0	0	0	0	1	1	0	0	1
•	0	0	0	0	0	0	0	1	1	1	0	1
▼	0	0	0	0	0	0	0	1	1	1	1	1
⊢	0	0	0	0	0	1	0	0	0	0	0	0

$T_{\gtrdot}$

FIG_2_2__ TABLEAUX BOOLEENS, ASSOCIES AUX RELATIONS $\lessdot$ ET $\lessdot$



(a)

Relations $\lessdot$, $\gtrdot$

(b)

Relations $\doteq$

FIG_2_3__ DIAGRAMMES CARTESIENS DES RELATIONS DE PRECEDENCE, DANS $\mathcal{T}_T$.

CHAPITRE 2

ANALYSE SYNTAXIQUE DE RECONNAISSANCE

GRAMMAIRE A PRECEDENCE D'OPERATEURS

	5	7	8	9	10	11
⊣	0	0	0	0	1	0
λ_2	0	0	0	0	0	0
λ_3	0	0	0	0	0	0
:=	0	0	0	1	0	0
λ_1	0	0	0	0	0	0
(	0	0	0	1	0	0
)	0	0	0	0	0	0
+	0	0	1	0	0	0
•	0	1	0	0	0	0
∨	0	0	0	0	0	0
⊢	0	0	0	0	0	0

T_{AG}

	5	7	8	9	10	11
5	1	0	0	0	0	0
7	0	1	0	0	0	0
8	0	1	1	0	0	0
9	0	1	1	1	0	0
10	1	0	0	0	1	0
11	0	0	0	0	0	1

T^{*}_{NG}

	⊣	λ_2	λ_3	:=	λ_1	(	)	+	•	∨	⊢
5	0	1	1	0	0	0	0	0	1	0	
7	0	1	0	0	1	1	0	0	1	0	
8	0	0	0	0	0	0	0	1	0	0	
9	0	0	0	0	0	0	1	0	0	0	
10	0	0	0	1	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	0	0	

T_{TG}

_FIG_2_4a_

TABLEAUX BOOLEENS, COMPOSANTS DU TABLEAU $T_{\lessdot}$, ASSOCIE A LA RELATION $\lessdot$

	5	7	8	9	10	11
⊣	0	0	0	0	0	0
λ_2	1	1	0	0	0	0
λ_3	1	0	0	0	0	0
:=	0	0	0	0	1	0
λ_1	0	1	0	0	0	0
(	0	0	0	0	0	0
)	0	1	0	0	0	0
+	0	0	0	1	0	0
•	0	0	1	0	0	0
∨	1	1	0	0	0	0
⊢	0	0	0	0	0	1

T^{*}_{TD}

	5	7	8	9	10	11
5	1	0	0	0	0	0
7	0	1	1	1	1	0
8	0	0	1	1	1	0
9	0	0	0	1	1	0
10	0	0	0	0	1	0
11	0	0	0	0	0	1

T^{**}_{ND}

	⊣	λ_2	λ_3	:=	λ_1	(	)	+	•	∨	⊢
5	0	0	0	1	0	0	0	0	1	0	
7	0	0	0	0	0	0	0	0	1	0	
8	0	0	0	0	0	0	0	1	0	0	
9	0	0	0	0	0	1	1	0	0	0	
10	0	0	0	0	0	0	0	0	0	1	
11	0	0	0	0	0	0	0	0	0	0	

T^{*}_{AD}

_FIG_2_4b_

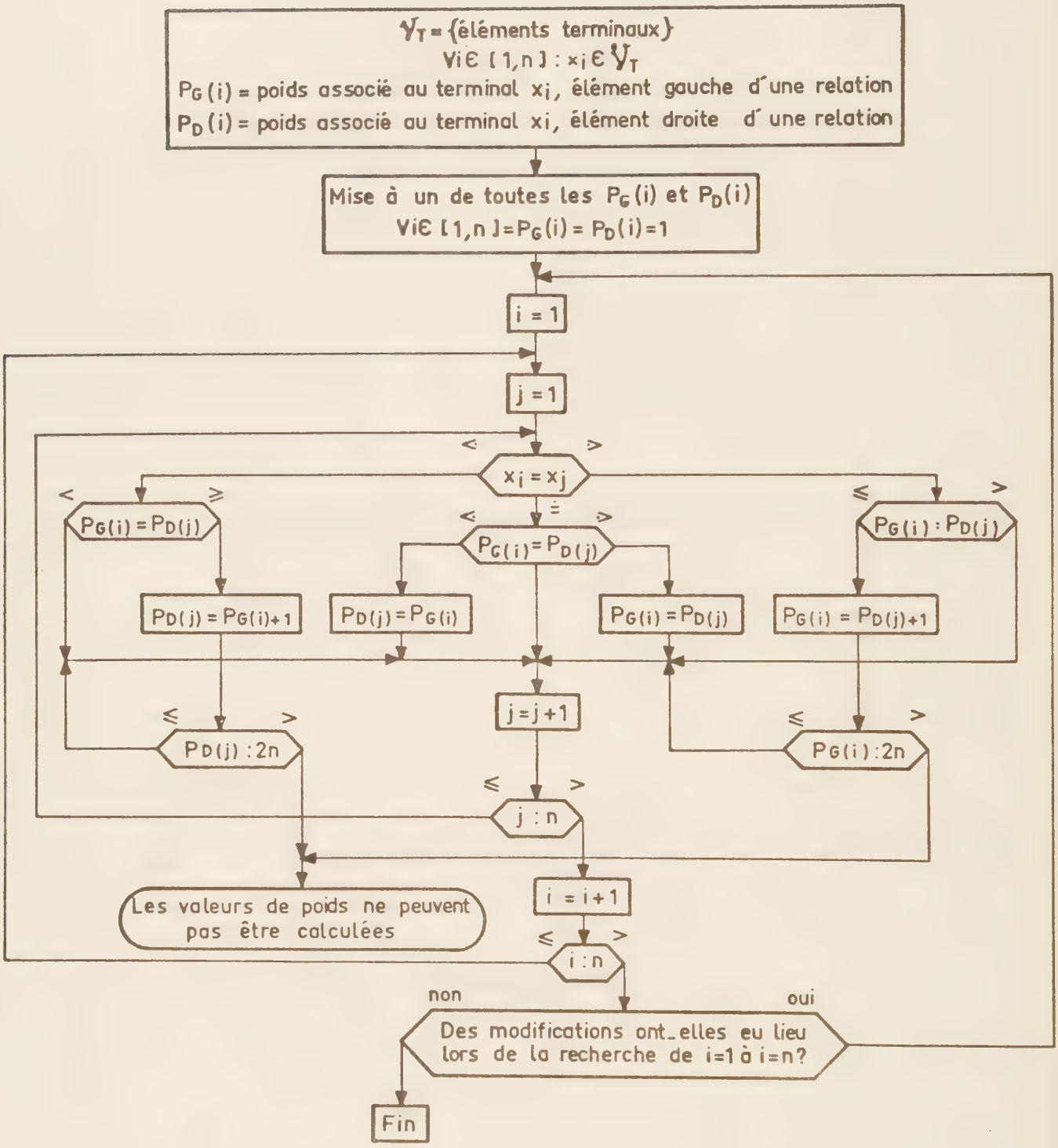
TABLEAUX BOOLEENS, COMPOSANTS DU TABLEAU $T_{\gtrdot}$, ASSOCIE A LA RELATION $\gtrdot$

	⊣	λ_2	λ_3	:=	λ_1	(	)	+	•	∨	⊢
⊣	$\lessdot$	$\lessdot$	$\lessdot$							$\lessdot$	$\doteq$
λ_2		$\gtrdot$				$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$	$\gtrdot$
λ_3		$\gtrdot$								$\gtrdot$	
:=	$\lessdot$		$\lessdot$	$\lessdot$		$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\lessdot$	$\gtrdot$
λ_1										$\gtrdot$	
(											
)										$\gtrdot$	
+										$\gtrdot$	$\gtrdot$
•										$\gtrdot$	$\gtrdot$
∨										$\gtrdot$	
⊢											

_FIG_2_5_

TABLEAU DES PRECEDENCES

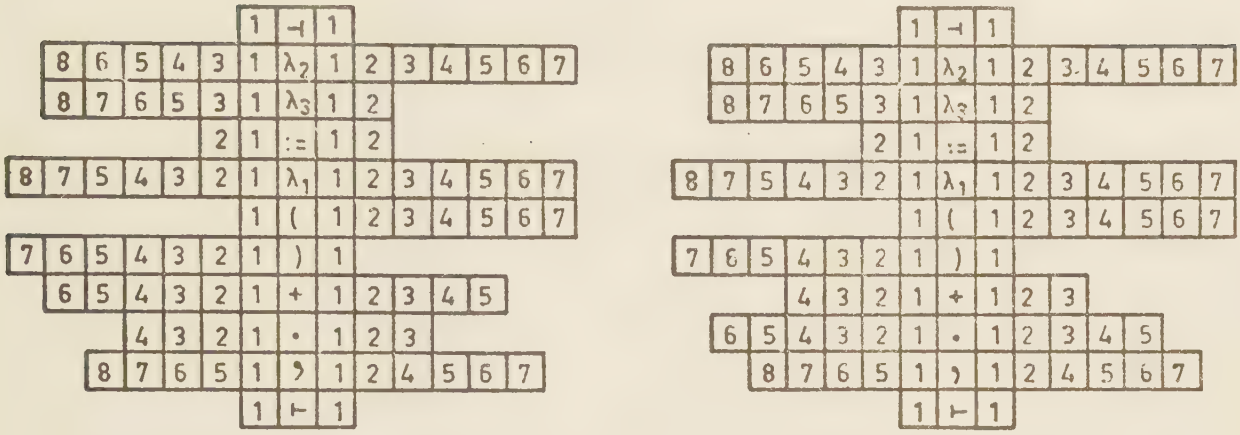
PROGRAMME CHOPIN
CHAPITRE 2
ANALYSE SYNTAXIQUE DE RECONNAISSANCE



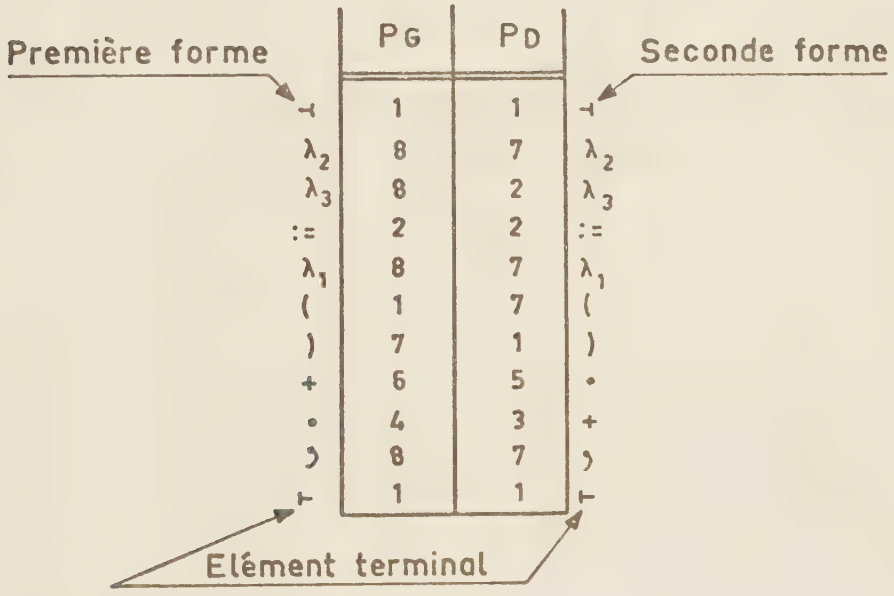
ALGORITHME DE RECHERCHE D'AFFECTATION DE POIDS

FIG. 2_6

PROGRAMME CHOPIN
CHAPITRE 2
ANALYSE SYNTAXIQUE DE RECONNAISSANCE



_a_ Evolution du tableau d'affectation des poids au cours de la recherche



_b_ Tableau d'affectation des poids. Grammaire CHOPIN

FIG. 2_7

PROGRAMME CHOPIN

CHAPITRE 2

ANALYSE SYNTAXIQUE DE RECONNAISSANCE

⊣	début de forme	implicite
λ_1	variable d'entrée	liste entrée en données
λ_2	variable secondaire	liste entrée en données
λ_3	variable de sortie	liste entrée en données
:=	affectation d'égalité	donnée d'entrée
<	affectation de borne inférieure	"
>	affectation de borne supérieure	"
+	opération de réunion	"
.	opération d'intersection	"
'	opération de complémentation	"
⊢	fin de forme	implicite

- vocabulaire terminal -

$V_N = \{ \langle \text{fonction} \rangle, \langle \text{primaire} \rangle, \langle \text{terme} \rangle, \langle \text{réunion} \rangle, \langle \text{expression} \rangle, S \}$

S est $\langle$première forme$\rangle$ pour une première forme
$\langle$seconde forme$\rangle$ pour une seconde forme

- vocabulaire non terminal -

$\langle \text{fonction} \rangle :: = \lambda_2 / \lambda_3 / \langle \text{fonction} \rangle'$

$\langle \text{primaire} \rangle :: = \lambda_1 / \lambda_2 / (\langle \text{terme} \rangle) / \langle \text{primaire} \rangle'$
$\langle \text{réunion} \rangle :: = \langle \text{primaire} \rangle / \langle \text{réunion} \rangle + \langle \text{primaire} \rangle$
$\langle \text{terme} \rangle :: = \langle \text{réunion} \rangle / \langle \text{terme} \rangle . \langle \text{réunion} \rangle$
$\langle \text{expression} \rangle :: = \langle \text{fonction} \rangle := \langle \text{terme} \rangle$
$\langle \text{seconde forme} \rangle :: = \dashv \langle \text{expression} \rangle \vdash$

$\langle \text{primaire} \rangle :: = \lambda_1 / \lambda_2 / (\langle \text{réunion} \rangle) / \langle \text{primaire} \rangle'$
$\langle \text{terme} \rangle :: = \langle \text{primaire} \rangle / \langle \text{terme} \rangle . \langle \text{primaire} \rangle$
$\langle \text{réunion} \rangle :: = \langle \text{terme} \rangle / \langle \text{réunion} \rangle + \langle \text{terme} \rangle$
$\langle \text{expression} \rangle :: = \langle \text{fonction} \rangle := \langle \text{réunion} \rangle$
$\langle \text{première forme} \rangle :: = \dashv \langle \text{expression} \rangle \vdash$

- liste des productions -

FIG. 2.8

CHAPITRE 2
ANALYSE SYNTAXIQUE DE RECONNAISSANCE
GRAMMAIRE CHOPIN

	1	2	3	4	5	6
⊣	0	0	0	0	1	0
λ_2	0	0	0	0	0	0
λ_3	0	0	0	0	0	0
:=	0	0	0	1	0	0
λ_1	0	0	0	0	0	0
(	0	0	0	1	0	0
)	0	0	0	0	0	0
+	0	1	0	0	0	0
•	0	0	1	0	0	0
,	0	0	0	0	0	0
⊢	0	0	0	0	0	0

T_{AG}

	1	2	3	4	5	6
1	1	0	0	0	0	0
2	0	1	0	0	0	0
3	0	1	1	0	0	0
4	0	1	1	1	0	0
5	1	0	0	0	1	0
6	0	0	0	0	0	1

T^{*}_{NG}

	⊣	λ_2	λ_3	:=	λ_1	(	)	+	•	,	⊢
1	0	1	1	0	0	0	0	0	0	1	0
2	0	1	0	0	1	1	0	0	0	1	0
3	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	0	0	1	0	0
5	0	0	0	1	0	0	0	0	0	0	0
6	1	0	0	0	0	0	0	0	0	0	0

T_{TG}

TABLEAUX BOOLÉENS COMPOSANTS DU TABLEAU T$\lessdot$
ASSOCIÉ A LA RELATION $\lessdot$

FIG_ 2.9 a

	1	2	3	4	5	6
⊣	0	0	0	0	0	0
λ_2	1	1	0	0	0	0
λ_3	1	0	0	0	0	0
:=	0	0	0	0	1	0
λ_1	0	1	0	0	0	0
(	0	0	0	0	0	0
)	0	1	0	0	0	0
+	0	0	1	0	0	0
•	0	0	0	1	0	0
,	1	1	0	0	0	0
⊢	0	0	0	0	0	1

$T^{\geqslant}_{TD}$

	1	2	3	4	5	6
1	1	0	0	0	0	0
2	0	1	1	1	1	0
3	0	0	1	1	1	0
4	0	0	0	1	1	0
5	0	0	0	0	1	0
6	0	0	0	0	0	1

$T^{*\geqslant}_{ND}$

	⊣	λ_2	λ_3	:=	λ_1	(	)	+	•	,	⊢
1	0	0	0	1	0	0	0	0	0	1	0
2	0	0	0	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	0	1	0	0	0
4	0	0	0	0	0	0	1	0	1	0	0
5	0	0	0	0	0	0	0	0	0	0	1
6	0	0	0	0	0	0	0	0	0	0	0

$T^{\geqslant}_{AD}$

TABLEAUX BOOLEENS COMPOSANTS DU TABLEAU T$\geqslant$
ASSOCIE A LA RELATION $\geqslant$

FIG_ 2.9 b

TABLEAU REDUIT DES PRECEDENCES CONDITIONNEES

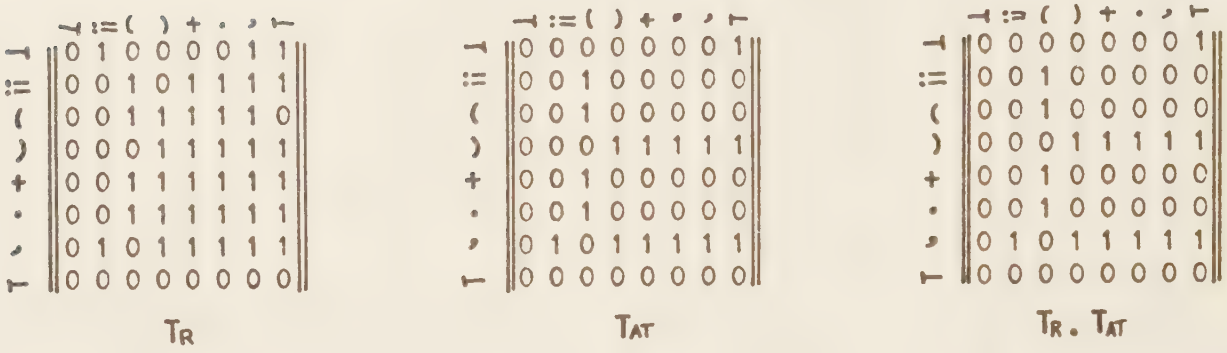


FIG_2.10 ___ ELEMENTS TERMINAUX ADJACENTS EN RELATION DE PRECEDENCE

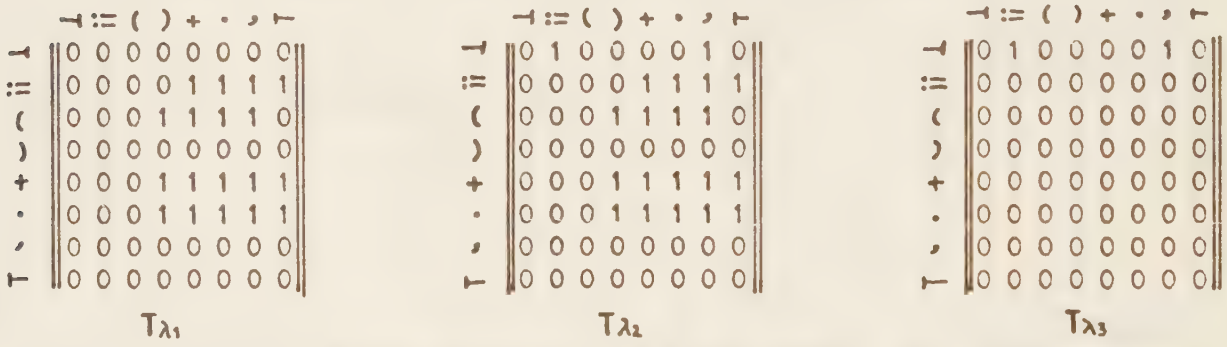


FIG_2_11 ___ ELEMENTS TERMINAUX, EN RELATION DE PRECEDENCE, SEPARES PAR UN IDENTIFICATEUR

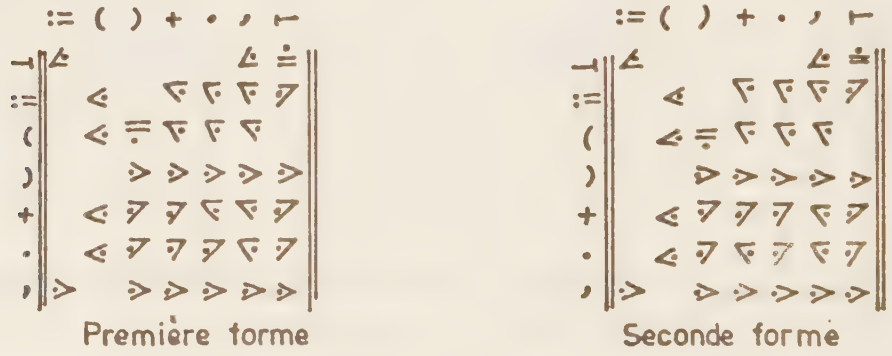


FIG_2_12 ___ TABLEAUX REDUITS DES PRECEDENCES CONDITIONNÉES

- CHAPITRE 3 -

ANALYSE SYNTAXIQUE DE REDUCTION

L'analyse de reconnaissance d'une expression booléenne permet de vérifier que son écriture est en conformité avec les règles imposées pour le type de représentation choisie.

L'expression doit maintenant être traitée et ramenée

- soit à une base complète (traitement en première forme),
- soit à une couverture complète (traitement en seconde forme),

lors de l'analyse syntaxique de réduction.

Les représentations en première et seconde formes sont des représentations dans deux ensembles, munis d'une structure de treillis booléen normal (cf. Annexes 7 et 8), duals. Les propriétés de l'un se transposent pour l'autre, les traitements sont semblables.

Nous allons donner quelques définitions qui nous permettent de traiter le problème d'une façon générale, le type de forme étant fixé. Puis nous rappellerons les méthodes générales de réduction. Nous situerons ces méthodes par rapport au cas qui nous préoccupe. Nous ferons apparaître la notion de forme allégée. Nous serons, alors, à même de caractériser la grammaire de réduction.

Nous avons représenté figure 3.4 un exemple d'expression booléenne et de traitement pour concrétiser les notions introduites dans ce chapitre.

3.1 GENERALITES

Les définitions et propriétés, énoncées ci-après, ne sont que l'adaptation à la grammaire CHOPIN des définitions plus générales, données en Annexe 7, relatives à la structure de treillis booléen normal :

$(\mathcal{B}, \cdot, +, ')$

3.11 Ensemble générateur de $\mathcal{B}$.

L'ensemble $\mathcal{G}$ générateur de $\mathcal{B}$ est le sous-ensemble de $\mathcal{V}_{10}$ (cf. 2.11) constitué par les seuls identificateurs de VENT et VSEC (cf. 2.11 et A1.1).

$$\mathcal{G} = X \cup Y$$

3.111 Variable booléenne

Une variable booléenne en forme normale est caractérisée par un identificateur.

Une variable booléenne en forme complémentée est un identificateur marqué du symbole de complémentation.

3.112 Fonction booléenne

Une fonction booléenne est une relation qui associe à un élément de $\mathcal{B}$ des éléments de $\mathcal{G}$. Elle prend ses valeurs dans l'ensemble Z (ou VSOR) ou Y (ou VSEC). Les valeurs sont appelées, improprement, fonctions booléennes et peuvent être représentées par les éléments Ψ_i de $\mathcal{B}$. Ceux-ci sont appelés représentations en forme normale (première ou deuxième).

3.12 Lois de composition

3.121 Loi de composition primaire

La loi de composition primaire est la loi de composition interne prioritaire :

- en première forme : l'intersection;
- en seconde forme : la réunion.

Elle est symbolisée par $\bullet$.

3.122 Loi de composition secondaire

La loi de composition secondaire est la seconde loi de composition interne :

- en première forme : la réunion ;
- en seconde forme : l'intersection.

Elle est symbolisée par $*$.

3.13 Racines

3.131 Définition

Une racine α est une composition primaire de variables booléennes y_i .

Une racine est racine d'une fonction booléenne si elle apparaît dans une représentation de cette fonction.

Une racine est génératrice d'une représentation en forme normale si elle est exprimée dans cette représentation.

3.132 Racine unité. Racine nulle

La racine unité η est l'élément neutre pour la loi de composition primaire.

La racine nulle N est l'élément neutre pour la loi de composition secondaire.

3.14 Radical

Un radical A est une composition secondaire de racines.

Comme une racine, un radical peut être dit : radical d'une fonction, radical générateur.

3.15 Relation d'ordre

Dans $\mathfrak{B}$, muni de la structure $(\mathfrak{B}, \circ , \dashv , ')$, il existe une relation d'ordre $\dashv$:

$$\forall A,B \in \mathfrak{B} : \; A \dashv B \rightleftharpoons A \circ B = B \; , \; A * B = A \qquad \langle A\text{-}31 \rangle$$

A est dit élément antérieur ou ascendant

B est dit élément postérieur ou descendant

$\forall A , B \in \mathfrak{B}; \exists C \in \mathfrak{B}: \; A \dashv C \dashv B$: B est descendant direct de C , A est ascendant direct de B.

Cette relation est l'implication en première forme, l'implication converse en seconde forme.

3.16 Complémentaires

Deux éléments A et B de $\mathcal{B}$ sont complémentaires si et seulement si :

$$A \circ B = N$$

$$A = B', \quad B = A'$$

$$A * B = \eta$$

3.17 Consensus

3.171 Définition

Le consensus α d'ordre 2 de deux racines β et γ, par rapport à la variable x, est l'antécédent direct, non antécédent de β et γ pris séparément, de $\beta * \gamma$:

$$\alpha = C_n(x; \beta, \gamma) \qquad \text{ou} \qquad \alpha = C_n(x; \gamma, \beta)$$

3.172 Racines adjacentes

Deux racines α et β sont adjacentes si elles admettent un consensus γ descendant direct commun :

$$\gamma = A_d(x; \alpha, \beta) \qquad \text{ou} \qquad \gamma = A_d(x; \beta, \alpha)$$

3.18 Racine première. Racine et radical de base

Une racine première est équivalente à :

- un implicant premier en première forme,
- une couverture première en seconde forme.

Une racine de base est équivalente à :

- une intersection de base en première forme,
- une réunion de base en seconde forme.

Le radical de base d'une fonction est le radical engendré par toutes les racines de base d'une fonction. La représentation est dite en forme canonique.

3.19 Représentation complète. Représentation première

Elles correspondent aux base et couverture complètes, premières, respectivement, des première et seconde formes. C'est-à-dire :

- au radical engendré par toutes les racines premières,
- à un radical réduit du précédent et irréductible,

respectivement.

3.2 LES DIFFERENTES METHODES DE REDUCTION

D'une manière générale, les différentes méthodes de réduction se répartissent en deux classes :

- dans la première (KARNAUGH, QUINE-Mc CLUSKEY, LEDLEY), le traitement est effectué à partir du radical de base de la fonction considérée ;

- dans la seconde (consensus itératifs, résidus), le traitement est effectué à partir d'une représentation où aucune génératrice n'est antérieure à une autre racine génératrice.

A cette différence près, ces méthodes comprennent deux phases, la dernière pouvant être commune :

- recherche de l'ensemble de toutes les racines premières de la fonction, donc de la représentation complète;

- recherche des racines essentielles et d'une représentation première.

3.21 Recherche des racines premières

Quelle que soit la méthode envisagée, la recherche des racines premières est basée sur les propriétés suivantes :

"tout consensus, d'ordre 2, de deux racines d'une fonction est racine de cette fonction" <C-A512>

"tout consensus engendré par une racine α , antérieure à une racine β , ne peut pas être racine première".

Cette dernière propriété est une conséquence du théorème A514.

3.211 Radical de base

L'algorithme utilisé est le suivant.

1/ - $p=n$ (n cardinal de $\mathcal{G}$);

2/ - recherche des consensus engendrés par adjacence entre les racines d'ordre p . Les consensus sont racines de degré $p-1$ de la fonction.

3/ - suppression des racines génératrices de ces consensus. Ces racines ne sont pas racines premières.

4/ - enregistrement des racines de degré p qui restent. Elles sont racines premières.

5/ - $p = p - 1$. Reprise en 2.

La recherche cesse lorqu'il n'existe aucun consensus obtenu par adjacence.

Les phases 1 et 2 se déduisent des propriétés énoncées plus haut.

La phase 4 se déduit de la définition des racines adjacentes et des propriétés de la relation d'ordre.

Les méthodes vont se différencier :

- soit par la systématisation : dans la méthode de KARNAUGH un choix heuristique est fait, alors qu'elle est automatique dans les autres;

- soit par le mode de représentation. La méthode de KARNAUGH est essentiellement graphique. Les autres méthodes sont basées sur des représentations numériques.

3.212 Consensus itératifs

L'algorithme de cette méthode est schématisée par la figure 3.6 :

1/ - $i = 1$

2/ - $j = i+1$

3/ - recherche du consensus d'ordre 2 engendré par la $i^{ème}$ racine avec la $j^{ème}$ racine.

4/ - suppression des racines antérieures à ce consensus, si il existe;

5/ - enregistrement du consensus, non antérieur à une racine, comme racine de la fonction;

6/ - la $(j+1)^{ème}$ racine existe et n'est pas consensus, d'ordre 2, engendré par la $i^{ème}$ racine :

- $j = j+1$
- retour en 3

- la $(j+1)^{ème}$ racine n'existe pas ou est consensus, d'ordre 2, engendré par la $i^{ème}$ racine :

- $i = i+1$
- retour en 2

7/ - arrêt avec la dernière racine.

3.213 Méthode des résidus

Les principes généraux sont les mêmes que ceux énoncés plus haut. La méthode de recherche se rapproche de celle des consensus itératifs, mais le processus de calcul des consensus est différent. Au lieu de rechercher les consensus engendrés par une racine donnée, on recherche les consensus relatifs à une variable biforme donnée.

L'algorithme est alors (cf. figure 3.2) :

1/ - $i := 1$

2/ - la $i^{ème}$ variable est biforme :

- 2.1/ - recherche des résidus par rapport à la variable en forme normale;
- 2.2/ - recherche des résidus par rapport à la variable en forme complémentée;
- 2.3/ - composition primaire des deux ensembles de résidus; les compositions non nulles sont les consensus relatifs à la $i^{ème}$ variable;
- 2.4/ - suppression des consensus, antérieurs à un autre consensus ou à une racine;
- 2.5/ - suppression des racines antérieures aux consensus qui restent;
- 2.6/ - enregistrement des consensus qui restent comme racines de la fonction;
- 2.7/ - $i = i + 1$, retour en 4

3/ - la $i^{ème}$ variable n'est pas biforme: retour en 2.7;

4/ - retour en 2 ou fin si la $i^{ème}$ variable n'existe pas.

3.22 Recherche d'une représentation première

Les méthodes sont diverses mais elles sont basées sur les propriétés suivantes :

- une racine est essentielle si elle n'est consensus, d'ordre 2, d'aucune racine première; <T - A 62>

- une racine est inessentielle si elle est consensus, d'ordre 2, de racines essentielles; <C - A 61>

- une racine est pseudo-inessentielle si elle est consensus, d'ordre 2, de racines essentielles et de racines pseudo-essentielles. <T-A 64> et <A - A 62>

Elles vont consister à

- rechercher les racines essentielles;
- supprimer les racines inessentielles;
- rendre éventuellement les racines non-essentielles, qui restent, racines pseudo-essentielles;
- supprimer les racines pseudo-inessentielles;

..... jusqu'à convergence.

Nous développerons, plus loin, l'algorithme pris pour le programme CHOPIN.

Les méthodes diffèrent par la présentation :

- méthode de recherche graphique (grille des implicants, ou couvertures, premières);
- méthode de recherche par expression littérale : chaque racine première est caractérisée par rapport aux autres racines premières. Seules, sont conservées les racines qui dépendent du plus petit nombre d'entre les autres racines.

3.3 DEVELOPPEMENT DES FORMES BRUTES

Que le traitement soit manuel ou mécanisé, le développement d'une expression en forme brute, en vue de l'obtention de sa représentation complète en forme normale, ne peut s'effectuer que par traitements successifs des parties de cette forme brute.

Ce traitement va avoir des caractéristiques différentes selon la forme choisie pour représenter ces parties (cf. Annexe 7).

Nous allons considérer les différents types et leur incidence sur le traitement. Le choix de la forme adéquate au traitement en découlera.

3.31 Représentation en forme canonique

Quelle que soit la partie d'expression à traiter, elle est mise sous forme canonique. Une racine α_i de degré p est représentée par le radical A_i engendré par les 2^{n-p} racines de base antérieures à α_i . Un radical B_j de racines génératrices α_{ij} est représenté par le radical A_j , engendré par les radicaux A_{ij} représentants en forme canonique des racines α_{ij} . Associons à ces représentations (cf. Annexe 7) les ensembles a_i, a_j, a_{ij} constitués par les racines de base y afférentes. Nous pouvons écrire :

$$\alpha_i \oplus \alpha_j \rightleftharpoons a_i \cap a_j$$
$$\alpha_i * \alpha_j \rightleftharpoons a_i \cup a_j$$
$$A_i \oplus A_j \rightleftharpoons a_i \cap a_j$$
$$A_i * A_j \rightleftharpoons a_i \cup a_j$$

Les lois, $\oplus$ et $*$, sont lois de composition interne, dans l'ensemble des formes canoniques. Cette propriété est intéressante, mais elle s'accompagne d'un nombre de contraintes qui sont prohibitives dans le cas d'emploi d'un ordinateur. Citons, entre autres :

- la nécessité de remplacer une racine quelconque, de degré p , par 2^{n-p} racines de base. L'encombrement mémoire nécessaire est plus important. Le temps de génération croît comme 2^{n-p} .

- lors de la composition de deux racines, ou de deux radicaux, le nombre de racines à composer est bien plus élevé. Certes, les opérations se traduisent par une recherche d'existence des racines de base, mais le nombre de recherches, dans le cas le plus défavorable, est élevé.

- l'encombrement minimum de réservation pour un radical est de 2^{n_M} , où n_M est le nombre maximum de variables que nous envisageons de traiter (nous avons choisi $n_M = 32$). Il y a toujours, au moins, deux radicaux à composer : il faut donc au minimum 2^{n_M+1} racines de base (deux racines unités peuvent être composées, c'est-à-dire 2×2^{n_M} racines de base);

- enfin la représentation en forme canonique n'est pas une fin en soi. Lorsque celle de l'expression traitée aura été obtenue, il faudra en extraire la forme complète.

3.32 Représentation en forme complète

Ici aussi, toute partie traitée de l'expression sera mise en forme complète. Mais les conditions de traitement vont quelque peu varier :

- une racine α_i est conservée telle quelle si elle est la partie considérée ;

- un radical, A_j , engendré par les racines α_{ij} , sera représenté par :

  - les racines α_{kj} qui n'admettent aucune autre racine postérieure;
  - les consensus, non racines antérieures, engendrés par ces racines α_{kj} .

- la composition de deux représentations en forme complète ne donne pas généralement une forme complète. Cette composition devra être traitée pour être transformée en sa représentation en forme complète.

3.33 Représentation en forme première

La représentation en forme première est une forme réduite de la forme complète. Elle pré-suppose l'existence de la forme complète et son traitement pour la transformer en forme première. Celle-ci a un nombre de racines premières plus faible que celle-là. Mais, cette réduction se traduit par un accroissement du temps de traitement, relativement à celui du cas précédent. D'autre part, comme pour les représentations en forme complète, les compositions de deux formes premières ne sont pas généralement des formes premières. Elles devront être suivies d'une recherche de forme complète, puis de sa réduction en forme première.

Le traitement de telles représentations est donc plus important que celui de représentations en forme complète.

3.34 Représentation choisie : la forme allégée

Il est évident que :

- la représentation en forme canonique n'est pas envisageable pour le traitement de formes brutes;

- la représentation en forme complète est plus intéressante que la représentation en forme première.

Cependant, elle risque de conduire à des temps de traitement importants.

Il est impensable de se contenter de développer les différentes parties puis de procéder à la recherche de la représentation complète. Celle-ci doit être précédée d'une réduction de la forme normale obtenue, en ne considérant que la réduction par la relation d'ordre $\dashv$. Or si nous envisageons l'algorithme, dans l'optique mécaniste, nous devons éviter de faire un premier examen d'une liste (ou pile), puis un nouvel examen.... etc. Il est souhaitable de grouper le maximum d'opérations faites à partir d'un même opérande. Ici nous devrions procéder, en même temps qu'à l'ordonnancement, à la recherche des consensus engendrés.. etc. Nous avons intérêt à disposer d'une représentation terminale ayant le moins de racines possibles.

Le traitement des représentations en forme allégée nous semble présenter un moyen terme. Une forme allégée (cf. Annexe 7) est une représentation dont aucune racine génératrice n'est antérieure à une autre racine génératrice, n'est adjacente à aucune autre racine génératrice et n'engendre aucun consensus, d'ordre 2, qui lui soit postérieur. Par rapport à la forme normale originelle, la forme allégée est une forme réduite. La représentation en forme complète fait intervenir un nombre plus élevé de racines (tous les consensus non antérieurs des racines génératrices) ou égal. Le traitement porte sur un nombre plus faible de racines, il sera moins long que celui d'une représentation en forme complète.

Mais ici encore, les compositions de formes allégées ne sont pas des formes allégées.

3.4 ANALYSE SYNTAXIQUE DE REDUCTION

L'analyse syntaxique de réduction est faite compte tenu de la grammaire, déjà définie par l'analyse de reconnaissance, mais aussi des règles de réduction propres à la nature du treillis booléen normal (cf. Annexe 7). C'est-à-dire que l'ensemble des productions est remplacé par l'ensemble des réductions (relations réciproques des précédentes) augmenté des réductions booléennes.

Avant de décrire l'organisation de cette analyse, nous allons rappeler les deux ensembles de réductions.

3.41 Ensemble des réductions syntaxiques

Cet ensemble se déduit facilement de l'ensemble des productions de la grammaire étudiée plus haut. Nous pouvons cependant caractériser un ensemble de réduction quelque peu différent qui donne les mêmes relations de précédence entre éléments terminaux.

L'analyse de reconnaissance étant faite, l'expression proposée est correctement construite. Il est inutile de procéder à une reconnaissance complète, la réduction ne porte que sur le second nombre, c'est-à-dire la représentation en forme normale. D'autre part, les variables secondaires et principales peuvent être groupées en variables proprement dites.

Nous écrirons donc :

<identificateur> / (<radical>) / <variable>′ → <variable (booléenne)>
<variable> / <racine> ⊕ <variable> → <racine>
<racine> / <radical> * <racine> → <radical>
:= <radical> → <forme complète>

Nous avons représenté figure 3.3 les différents tableaux booléens et le tableau de précédence déduits de cet ensemble de réduction. Le tableau de précédence est semblable à celui obtenu au chapitre 2.

3.42 Réductions des chaines terminales

Les différentes réductions des chaînes terminales se déduisent des propriétés du treillis booléen normal. Nous allons les énumérer, en nous référant éventuellement à ces propriétés, démontrées ou établies en Annexes 6, 7 et 8.

y_i → <variable (booléenne)>
α_i → <racine>
A_i → <radical>
$\underline{A_i}$ → <forme complète>
N → <racine nulle>
η → <racine unité>
$y_i \oplus y_i \to y_i$
$y_i \oplus y'_i \to N$
$y_i * y'_i \to \eta$

$\alpha_i = \bigoplus_{j=1}^{p} y_{ij}$ $\quad \alpha_k = \bigoplus_{l=1}^{m} y_{kl}$

$\forall j \in [1,p] : \exists l \in [1,m]; y_{ij} = y_{kl} : \alpha_k \dashv \alpha_j$

$\alpha_k \oplus \alpha_j \to \alpha_k$

$\alpha_k * \alpha_j \to \alpha_j$

$\exists j \in [1,p], l \in [1,m]; y_{ij} = y'_{kl} : \alpha_k \oplus \alpha_j \to N$

$\exists! j \in [1,p], \exists! l \in [1,m]; y_{ij} = y_{kl} : \beta = C_m(y_{ij}; \alpha_i, \alpha_j)$

$\alpha_i \dashv \beta : \alpha_i * \alpha_j \to \beta * \alpha_j$

$\alpha_i \dashv \beta, \alpha_j \dashv \beta : \alpha_i * \alpha_j \to \beta$

$A_i \oplus N \to N$
$A_i * \eta \to \eta$
$A_i \oplus \eta \to A_i$
$A_i * N \to A_i$

Ces différentes réductions sont utilisées pour le calcul de la représentation en forme allégée. Il faut leur ajouter les suivantes pour la recherche de la forme complète et de la forme première :

$$A_i = \overset{p}{\underset{j=1}{*}} \alpha_j \overset{m}{\underset{k=1}{*}} \beta_k \; ;$$

$$\forall j \in [1,p], \forall k \in [1,m] : \alpha_j \neq \beta_k \, , \; \beta_k \neq \alpha_j$$

$$\forall k \in [1,m] ; \exists (h,\ell) \in [1,p]^2 ; \; \beta_k = C_n(y; \alpha_h, \alpha_\ell)$$

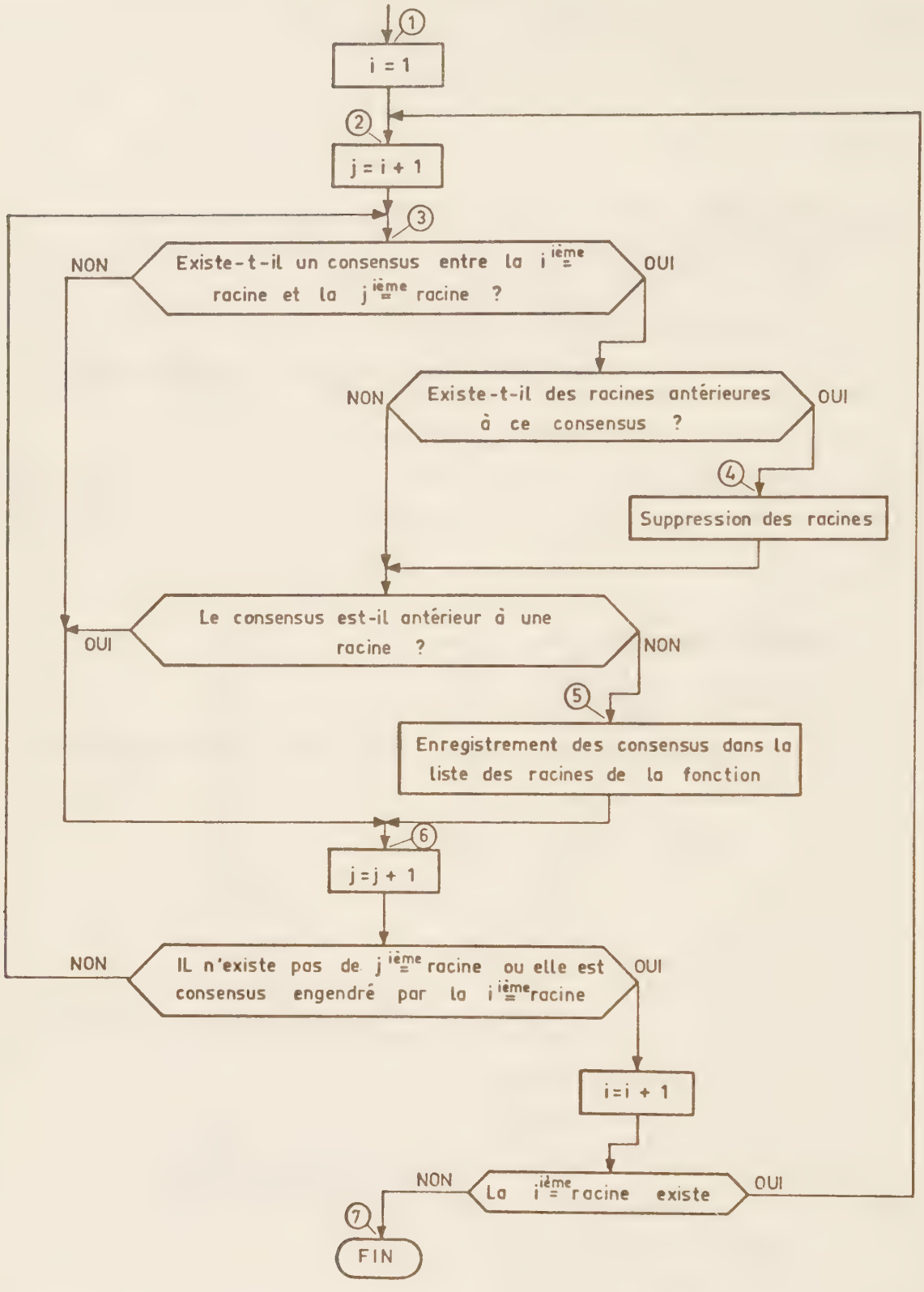
$A_i \longrightarrow$ <forme complète>

$$A_i = \overset{p}{\underset{j=1}{*}} \alpha_j$$

$$\forall j,k \in [1,p] : \alpha_j \neq \alpha_k$$

$$\forall j \in [1,p] : \alpha_j \neq \overset{j-1}{\underset{k=1}{*}} \alpha_k \overset{p}{\underset{k=j+1}{*}} \alpha_k$$

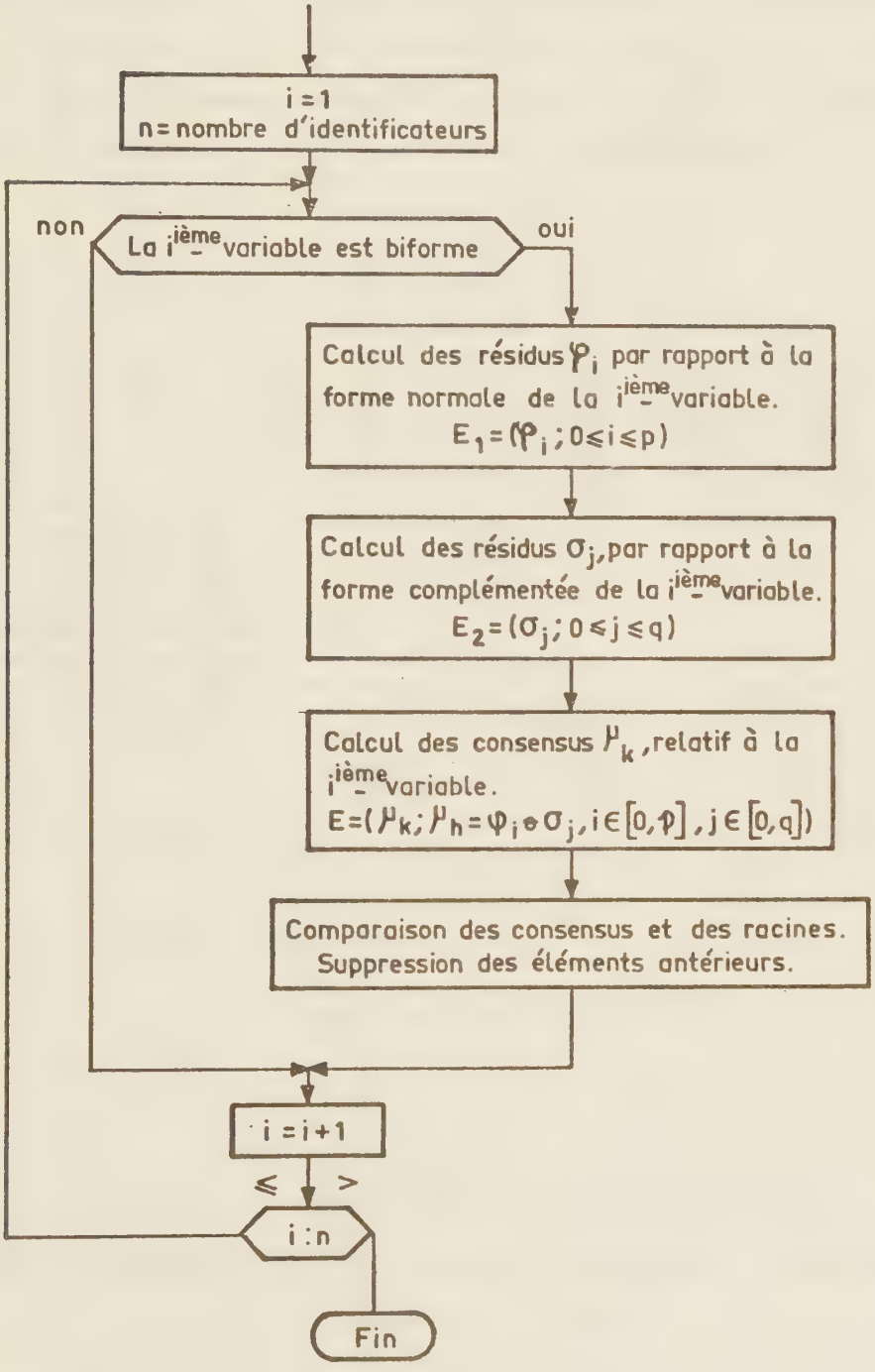
$A_i \longrightarrow$ <forme première>



RECHERCHE DES RACINES PREMIERES PAR LA METHODE DES CONSENSUS ITERATIFS

FIG_ 3_1

PROGRAMME CHOPIN
CHAPITRE 3
ANALYSE SYNTAXIQUE DE REDUCTION



RECHERCHE DES RACINES PREMIERES PAR LA METHODE DES RESIDUS.

FIG.3_2

PROGRAMME CHOPIN

CHAPITRE 3

ANALYSE SYNTAXIQUE DE REDUCTION

1 <identificateur>/(<radical>)/<variable> ⟶ <variable (booléenne)>

2 <variable>/<racine> ⊖ <variable> ⟶ <racine>

3 <racine>/<radical> * <racine> ⟶ <radical>

4 := <radical> ⊢ ⟶ <forme complète>

T_{AG}

	1	2	3	4
:=	0	0	1	0
(	0	0	1	0
)	0	0	0	0
*	0	1	0	0
⊖	1	0	0	0
,	0	0	0	0
⊢	0	0	0	0

T^{*}_{NG}

	1	2	3	4
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	0	0	0	1

T_{TG}

	:=	(	)	*	⊖	,	⊢
:=	0	1	0	0	0	1	0
(	0	0	0	0	1	0	0
)	0	0	0	1	0	0	0
*	1	0	0	0	0	0	0

T^{*}_{TD}

	1	2	3	4
:=	0	0	0	0
(	0	0	0	0
)	1	0	0	0
*	0	0	1	0
⊖	0	1	0	0
,	1	0	0	0
⊢	0	0	0	1

T^{*x}_{ND}

	1	2	3	4
1	1	1	1	0
2	0	1	1	0
3	0	0	1	0
4	0	0	0	1

T^{x}_{AD}

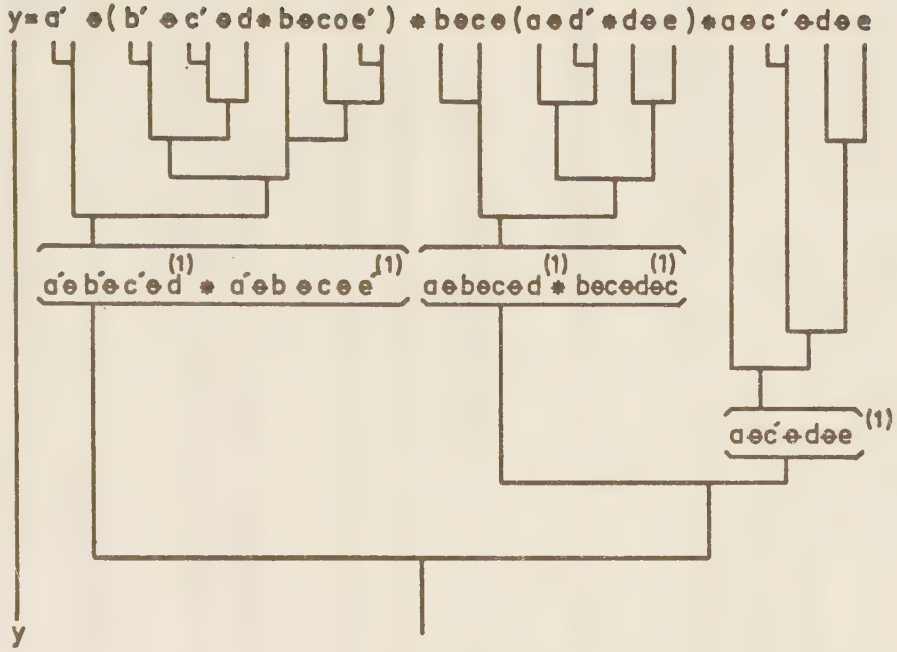
	:=	(	)	*	⊖	,	⊢
1	0	0	0	0	0	1	0
2	0	0	0	0	1	0	0
3	0	0	1	1	0	0	1
4	0	0	0	0	0	0	0

	:=	(	)	*	⊖	,	⊢
:=	◁		◁	◁	◁		
(	◁	≐	◁	◁	◁		
)		▷	▷	▷	▷	▷	▷
*	◁	▷	▷	◁	◁	▷	
⊖	◁	▷	▷	▷	◁	▷	
,		▷	▷	▷	▷	▷	
⊢							

ENSEMBLE DE REDUCTIONS

(TABLEAUX BOOLEENS, TABLEAU DES PRECEDENCES ASSOCIE)

FIG. 3.3



(identificateurs) = (y,a,b,c,d,e)

variables booléennes en forme complémentée

radicaux

radicaux générateurs de la représentation

racines génératrices de la représentation

radical générateur

radical représentation allégée
"fonction booléenne" en forme normale

IDENTIFICATEUR, VARIABLES, FONCTION, RACINES, RADICAUX :

EXEMPLE.

FIG. 3 _ 4

SECONDE PARTIE

DESCRIPTION DU PROGRAMME CHOPIN

- CHAPITRE 4 -

ECONOMIE ET ORGANISATION GENERALE

Deux tâches sont dévolues au programme CHOPIN :

- analyse de reconnaissance de la représentation en forme brute de l'expression présentée;

- réduction de cette expression en une représentation complète et une représentation première.

La première ne peut être accomplie que si :

- la grammaire est définie,

- le vocabulaire terminal, à partir duquel le langage doit être engendré, est donné.

Deux grammaires :

- l'une qui permet de construire des expressions en première forme (réunion d'intersections);

- l'autre qui permet la construction des expressions en seconde forme (intersection de réunions);

sont mises à la disposition de l'utilisateur qui pourra choisir l'une d'entre elles.

Par contre, l'utilisateur est maître du langage employé. Il lui est possible d'introduire le vocabulaire terminal qu'il souhaite et, éventuellement, de le modifier.

L'analyse de reconnaissance terminée, l'expression correctement codée et représentée, la seconde tâche peut alors être entreprise. Son exécution consiste en deux phases :

- allégement de l'expression,

- construction de la représentation complète dont une partie, nettement dissociée, est une représentation première possible.

Ces deux tâches peuvent être répétées à volonté avec :

- le même langage, conservation des résultats acquis (en vue d'une recherche ultérieure : construction de matrices séquentielles) ou non;

- le même vocabulaire terminal mais une autre grammaire, conservation ou effacement des précédents résultats;

- un vocabulaire terminal différent, totalement ou partiellement.

La figure 4.1 représente l'organisation générale du programme CHOPIN avec l'indication des sous-programme concernés. La figure 4.2 reproduit la liste alphabétique des différents sous-programmes, leurs fonctions et l'origine de leur nom ou sigle.

Les différents programmes et sous-programmes sont décrits dans la troisième partie.

4.1 INITIALISATION

L'initialisation consiste, en plus des inévitables déclarations de types et de dimensions que nous retrouvons dans tout programme ou sous-programme, en :

- l'introduction de la matrice réduite des précédences conditionnées (cf. 2.32) qui caractérise les relations de précédence, définies par les grammaires;

- l'enregistrement des identificateurs de types de cartes, de types de variables, des opérateurs et des types de décisions;

- l'affectation des dimensions maxima à certaines listes ou à certains tableaux;

- le calcul de certains paramètres :

  - puissances entières de 2^n $(0 \leqslant n \leqslant 31)$. Le calcul est effectué par appel du sous-programme général : PUISSC;
  - nombres de 1, entrant dans la représentation des nombres binaires dont la valeur est comprise entre 0 et 256. Ce calcul est effectué lors de l'exécution du sous-programme général : HARMON.

Cette première séquence du traitement est commandée

- soit par le programme : BLOCK DATA,
- soit par le programme principal : MAIN.

4.2 CONSTITUTION DU VOCABULAIRE TERMINAL

Les grammaires sont spécifiées, les différentes valeurs des paramètres utiles à la poursuite de l'exécution initialisées ou calculées. Le langage peut être caractérisé.

Pour une première exécution, les données en entrée sont alors, obligatoirement, les spécifications :

- des identificateurs des variables (ou générateurs de l'ensemble des expressions),

- et des symboles opératoires.

Pour les exécutions suivantes, elles seront limitées à celles qui sont nécessaires (cf. 5.3 et Annexe 10).

Les données doivent être introduites dans les limites d'un même paquet, mais dans un ordre quelconque. Elles sont traitées par l'intermédiaire du sous-programme principal, REPART, des sous-programmes généraux, PRELUD, DECHIF, ARPEGE, et des sous-programmes auxiliaires, OPERA et INSTRU.

Une liste de symboles avec leur affectation aux signes opératoires, une liste des variables réparties selon leur type (entrée, secondaire, secondaire-sortie, sortie) sont constituées. Les variables sont simultanément codées, en vue de leur utilisation ultérieure pour le traitement de l' (ou des) expression, leurs caractéristiques (nombre de caractères alpha-numériques de l'identificateur, adresses, poids ...) enregistrées.

La fin de cette séquence du traitement est marquée par l'impression de ces deux listes d'éléments du vocabulaire terminal.

4.3 ANALYSE DE RECONNAISSANCE

La partie la plus importante, quant à ses applications, peut être maintenant abordée. Elle aboutira à la représentation réduite de l'expression. Le sous-programme principal TRADUC est conçu pour organiser cette seconde partie du traitement.

Les données en entrée traitées maintenant sont celles qui concernent les expressions, c'est-à-dire les expressions littérales et les délimiteurs de paquets (fin de paquet, demande de modifications).

Pour toutes les expressions écrites à partir d'un même vocabulaire terminal, le cycle analyse-réduction sera le même. Mais il pourra se faire selon l'une ou l'autre des grammaires. Le choix de celle-ci est fait au moment de l'analyse de la carte (le type de grammaire choisie est marqué avec l'expression). Le format des données en entrée est décrit avec le sous-programme PRELUD.

L'analyse de chaque expression est entreprise après que le type de grammaire ait été reconnu. Elle conduit :

- soit à une expression codée, enregistrée dans le tableau EQUATI et le tableau SIGLOG, puis à la poursuite du traitement;

- soit à un arrêt du traitement de l'expression, l'impression du diagnostic d'erreur, la poursuite de l'exécution avec l'expression suivante ou l'arrêt du traitement.

Le sous-programme secondaire MVMNTS, avec les sous-programmes auxiliaires ACCORD et SOLFEJ, assure cette analyse de reconnaissance et le codage de l'expression brute ou originelle.

L'organisation de la séquence d'allégement est alors confiée au sous-programme secondaire EXECUT.

4.4 ANALYSE DE REDUCTION

4.41 Allégement

Compte tenu du langage choisi, les différentes opérations vont être effectuées à partir des données codées de la représentation brute de l'expression.

Les limites des propositions sont déterminées par la recherche des relations de précédence entre éléments terminaux, autres qu'identificateurs de variables. Ces éléments sont enregistrés dans le tableau SIGLOG.

Les phrases sont alors construites sous formes réduites, allégées. Cette construction peut prendre différentes formes selon les phrases "opérandes". Si nous symbolisons par ⊖ l'opération prioritaire (intersection en première forme, réunion en seconde forme), par * la seconde opération, les différentes constructions sont :

- ⊖ entre deux racines. Le sous-programme DUO est appelé.
- ⊖ entre une racine et un radical : sous-programme auxiliaire TRANS1;

- ⊖ entre deux radicaux : sous-programme auxiliaire TRANS2;
- * entre deux racines : sous-programme auxiliaire PASDE2;
- * entre une racine et un radical : sous-programme auxiliaire PASDE2;
- * entre deux radicaux : sous-programme auxiliaire POLKA.

L'appel du sous-programme adéquat est fait par l'intermédiaire du sous-programme CHORAL. En retour, celui-ci caractérise le résultat: racine ou radical.

Lorsque les opérations d'allégement sont terminées, la représentation allégée de l'expression, sauf lorsqu'elle est triviale ou qu'elle se réduit à une racine, est disponible dans le tableau EXPRES.

4.42 Réduction

La dernière séquence du traitement est maintenant confiée au sous-programme : NOCTUR.

La réduction de l'expression allégée s'effectue en deux étapes, sous le contrôle du sous-programme : NOCTUR.

Au cours de la première, la liste des racines de l'expression allégée est réarrangée, pour faciliter la recherche ultérieure des consensus, selon la méthode des résidus (cf. Annexe 8). Ces consensus sont alors calculés, les racines partagées en deux listes :

- essentielles
- et non-essentielles proprement dites.

L'opération est répétée jusqu'au moment où le dernier élément générateur biforme a été considéré.

La recherche des racines essentielles, ou pseudo-essentielles, fait suite. Elle constitue la seconde étape de la réduction.

Cette recherche et cette réorganisation de la représentation sont faites avec l'aide du sous-programme secondaire : BARCAR.

Les racines de la représentation complète sont enregistrées dans le tableau EXPRES en deux listes dont la première constitue la représentation première obtenue.

4.5 IMPRESSION

4.51 Impression de la rerrésentation obtenue

Cette séquence du traitement consiste en :

- un transcodage du langage machine en celui caractérisé en entrée;

- une impression de la représentation littérale obtenue. Celle-ci se décompose en deux parties. La première est la représentation de la forme première. La seconde est un addendum, constitué par les racines premières qui doivent être ajoutées aux précédentes pour former la représentation littérale complète.

4.52 Impression du diagnostic d'erreur

Selon l'erreur décelée, le traitement se poursuit ou non, le sous-programme général COUACS assure l'impression du diagnostic. La décision de poursuite, ou d'arrêt du traitement, est confiée au sous-programme principal appelant.

4.6 POURSUITE DE L'EXECUTION

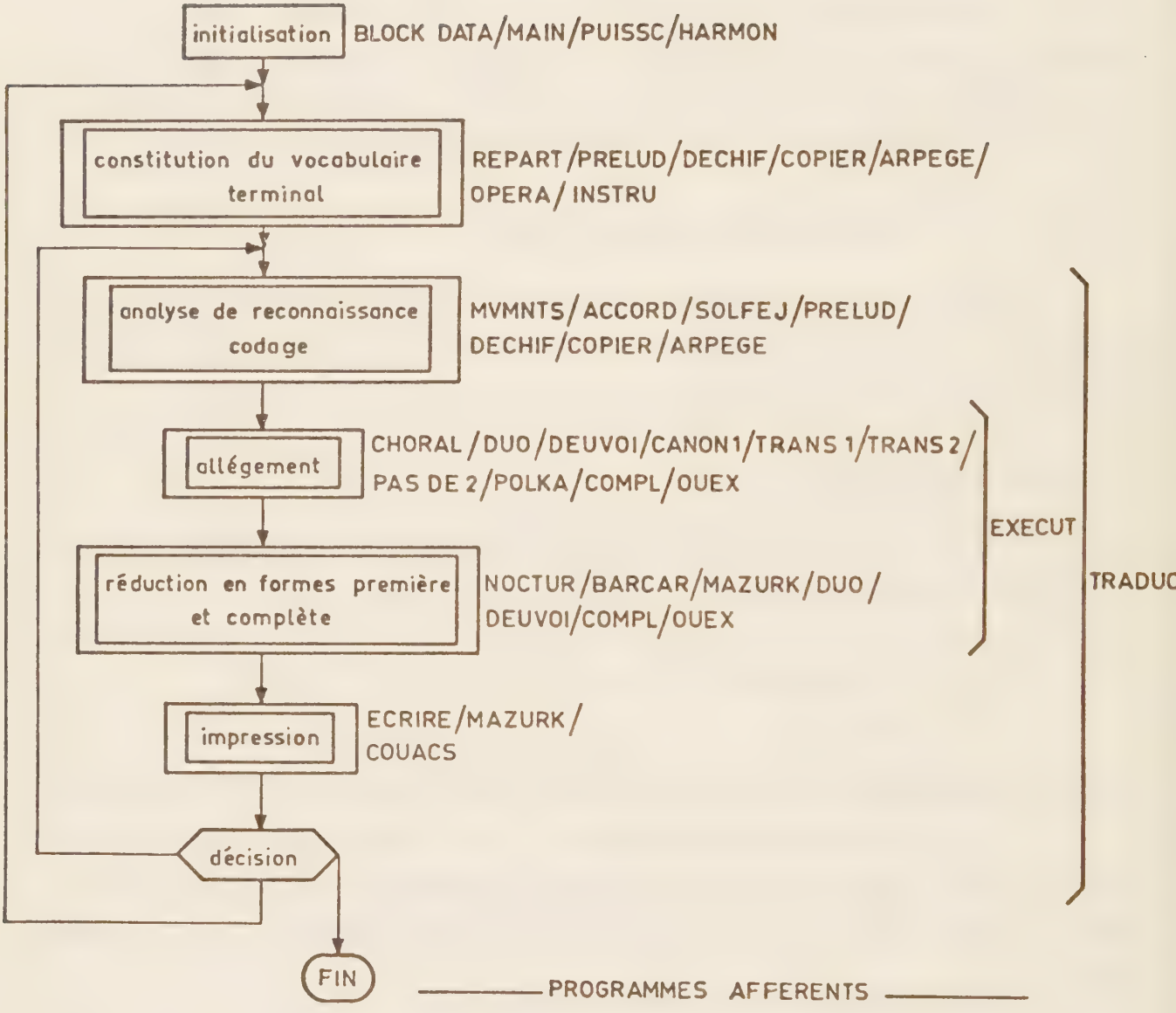
Une carte de choix délimite chaque paquet de cartes de données. Le traitement sera poursuivi, arrêté ou repris au point choisi :

- traitement de l'expression suivante;
- modification partielle ou totale du vocabulaire terminal;
- fin d'exécution.

La décision incombe au programme principal MAIN pour les deux derniers cas.

Lorsque les résultats sont conservés, ils restent disponibles dans le tableau EXPRES, et leurs caractéristiques (premier membre, séparateur de membres, type de langage, adresses dans le tableau EXPRES) sont enregistrées dans le tableau PAREQU.

PROGRAMME CHOPIN
CHAPITRE 4
ECONOMIE ET ORGANISATION GENERALE



ORGANIGRAMME FONCTIONNEL

FIG. 4_1

Nom du programme	Fonction	Origine du nom
ACCORD	Reconnaissance d'identificateur - Codage numérique de la variable	ACCORD
ARPEGE	Concaténation des caractères d'un mot	ARPEGE
BARCAR	Etablissement de la représentation première	BARCARolle
BLOCK DATA	Initialisation générale	BLOCK DATA
CANON1	Recherche d'allègement entre deux racines	CANON à 1 voie
CHORAL	Organisation de l'allègement, appel des sous-programmes d'allègement	CHORALe
COMPL	Complémentation bit à bit	COMPLémentation
COPIER	Entrée secondaire de DECHIF	COPIER
COUACS	Impression des diagnostics d'erreurs	COUACS
DECHIF	Recherche du type du caractère lu ou significatif, commande de concaténation	DECHIFfrer
DEUVOI	Entrée secondaire de DUO - Recherche des générateurs biformes	DEUx VOIes
DUO	Constitution des racines, recherche des générateurs biformes	DUO
ECRIRE	Impression des expressions	ECRIRE
EXECUT	Organisation de la réduction des expressions	EXECUTion
HARMON	Calcul du nombre de uns des 256 premiers nombres binaires	HARMONic
INSTRU	Constitution de la liste des variables	INSTRUmentation
MAIN	Initialisations particulières, appels des sous-programmes principaux, organisation générale du traitement	MAIN
MAZURK	Calcul du degré de la racine	MAZURKa
MVMNTS	Analyse syntaxique de reconnaissance	MouVeMeNTS
NOCTUR	Préparation de la représentation pour la recherche de la représentation première	NOCTURne
OPERA	Constitution de la liste de symboles opératoires	OPERA
OUEX	Opération logique OU-EXclusif	OU-EXclusif
PASDE2	Composition d'une racine et d'un radical, de deux radicaux	PAS DE 2
POLKA	Entrée secondaire de PASDE2, composition de deux radicaux	POLKA
PRELUD	Recherche du type de la carte lue	PRELUDe
PUISSC	Calcul des puissances entières de 2	PUISSanCes de 2
REPART	Enregistrement des données principales	REPARTition
SOLFEJ	Reconnaissance et codage du symbole opératoire	SOLFEge
TRADUC	Initialisation, appel des sous-programmes principaux de la réduction	TRADUCtion
TRANS1	Intersection (ou réunion) d'une racine et d'un radical, de deux radicaux	TRANSposition 1
TRANS2	Entrée secondaire de TRANS1, cas de deux radicaux	TRANSposition 2

FONTIONS DES DIFFERENTS SOUS_PROGRAMMES
FIG. 4_2

- CHAPITRE 5 -

CONSTITUTION DU VOCABULAIRE TERMINAL

Le vocabulaire terminal est constitué des différents éléments utilisés lors de l'écriture des expressions logiques :

- identificateurs de variables,
- symboles opératoires ou de ponctuation.

Alors que les symboles sont en nombre limité et leur type imposé, les identificateurs de variables, dans les limites admissibles, peuvent être en nombre quelconque. Cependant, ils doivent se répartir, qualitativement, selon les rôles que les variables ont à jouer (cf. Annexe 5). Deux des catégories doivent, impérativement, être caractérisées :

- les variables d'entrée pour des raisons évidentes;
- les variables secondaires ou les variables de sortie.

Aucun symbole, ou identificateur, n'est donné implicitement. Il faut donc, au début d'une première exécution du programme CHOPIN, que les opérateurs et les identificateurs, choisis par l'utilisateur, soient présentés.

Par la suite, ils pourront être modifiés, en totalité ou en partie :

- des opérateurs peuvent être ajoutés, ou remplacés, dans la liste originelle;
- les listes d'identificateurs peuvent être reconstituées. Elles devront être introduites comme pour une première exécution.

Ces deux types de modifications du vocabulaire terminal sont indépendantes.

Les données sont présentées à l'ordinateur sous forme de cartes, selon un format voisin du format FORTRAN (cf. description du sous-programme PRELUD).

5.1 LES DIFFERENTS ELEMENTS DU VOCABULAIRE TERMINAL

5.11 Symboles opératoires. Ponctuation

Pour faciliter leur reconnaissance, les symboles sont identifiés. L'utilisateur accompagnera les symboles, qu'il désire utiliser, de leur identificateur (cf. OPERA).

Au nombre de huit, ils se répartissent en trois catégories.

5.111 Symboles opératoires

Ils correspondent aux signes opératoires logiques :

- ET,
- OU,
- PAS

5.112 Signes de ponctuation, de groupements

Ce sont en fait les parenthèses classiques, gauche (PARG) et droite (PARD). L'utilisateur a la possibilité de les remplacer par tout autre symbole qu'il désire.

5.113 Séparateurs de membres

Les expressions introduites peuvent être de différents types :

- représentation d'une fonction complètement définie,
- représentation d'une fonction incomplètement spécifiée.

Pour le premier type, le symbole identifié EGAL convient parfaitement.

Pour le second type, deux symboles, identifiés LINF (Limite INFérieure) et LSUP (Limite SUPérieure), lui sont adjoints :

- EGAL est utilisé lorsque la fonction incomplètement spécifiée, est représentée sous sa forme normale et sous sa forme complémentée;

- LINF et LSUP sépareront, respectivement, les deux membres de la représentation de la borne inférieure et ceux de la représentation de la borne supérieure. Lorsque l'identificateur choisi pour caractériser ces bornes est différent, de tels séparateurs sont inutiles.

5.12 Identificateurs de variables

Dans le cadre des limites qui est fixé pour leur représentation, l'utilisateur dispose d'une grande liberté pour le choix des identificateurs.

Le type de variables est spécifié par l'inscription, dans la zone prévue à cet effet, de l'un des sigles :

- VENT pour les variables d'entrée,
- VSEC pour les variables secondaires,
- VSOR pour les variables de sorties.

L'utilisateur veillera :

- à ne pas grouper sur une même carte des identificateurs de variables de types différents,
- à caractériser la première carte d'une suite,
- à ne pas employer le même identificateur pour une variable d'entrée et une variable secondaire ou de sortie, simultanément. Il lui est cependant loisible de caractériser une même variable : variable de sortie et variable secondaire, ou inversement.

Il a toute liberté pour répartir les identificateurs sur différentes cartes, formant une suite ou non. Dans le premier cas, seule la carte de tête doit spécifier le type de variables.

La notice d'exploitation, décrite en Annexe 10, détaille les formats, contraintes ou libertés, auxquels l'utilisateur doit se conformer.

5.13 Présentation de la liste des éléments en entrée

L'ordre de présentation des différentes cartes est libre. Le programme procédera à leur réarrangement éventuel.

La fin du paquet doit seulement être marquée par une carte "FIN".

5.2 CONSTITUTION DU VOCABULAIRE TERMINAL

Commandée par le programme principal MAIN, la recherche des éléments du vocabulaire terminal a lieu :

- soit en début d'exécution,
- soit après la réduction d'expression. Elle correspond alors a une modification du vocabulaire (v. plus loin).

Sur la figure 5.1, l'organigramme fonctionnel de cette partie du traitement est schématisé.

Le sous-programme principal REPART contrôle l'ensemble des opérations nécessaires à la constitution de ce vocabulaire. En particulier, il vérifie la conformité des types de données présentées avec la catégorie des modifications demandées.

La lecture des cartes, la détermination de leur type est faite sous le contrôle du sous-programme général : PRELUD.

Lorsqu'une carte "données" est reconnue, le sous-programme REPART recherche le type de données : Opérateurs ou Variables. Il appelle le sous-programme auxiliaire correspondant : OPERA ou INSTRU.

5.21 Symboles opératoires

Avec l'aide des sous-programmes généraux, DECHIF, ARPEGE, le sous-programme OPERA reconstitue l'identificateur. Il affecte le symbole présenté à l'opérateur correspondant. Aucune vérification d'affectation préalable ou d'existence de symbole identique n'est faite.

Lorsque la carte a été entièrement explorée, le retour vers le sous-programme REPART a lieu.

5.22 Variables

La catégorie des variables est déterminée à l'aide du sous-programme : INSTRU. Assisté des sous-programmes, DECHIF et ARPEGE, il va reconstituer chacun des identificateurs. Il recherche alors la non-existence d'un identificateur semblable. Dans le cas contraire :

- si la variable est variable d'entrée : l'exécution est arrêtée;
- sinon, la variable est classée variable secondaire dédoublée.

L'exécution se poursuit par l'enregistrement de l'identificateur dans la liste correspondante et à la suite. Le nombre de caractères alpha-numériques de l'identificateur est également enregistré.

Lorsque la carte a été entièrement lue, le retour vers le sous-programme REPART s'effectue.

5.23 Impression des éléments du vocabulaire terminal

Le vocabulaire terminal est ainsi progressivement constitué ou modifié.

La détection d'une carte "FIN" par le sous-programme PRELUD provoque la commande par le sous-programme REPART de

- la remise en ordre de la liste des identificateurs de variables. Les identificateurs sont alors repérés, en commençant par les identificateurs des variables d'entrée puis les variables secondaires, secondaires dédoublées et enfin les variables de sortie. Leurs adresses sont enregistrées dans un tableau POIVAR, et un poids égal à 2^{n-1} (n : rang de l'identificateur dans la liste) est alloué à chacun d'eux.

- l'impression de la liste des éléments du vocabulaire terminal, avec indication du type pour les identificateurs de variables.

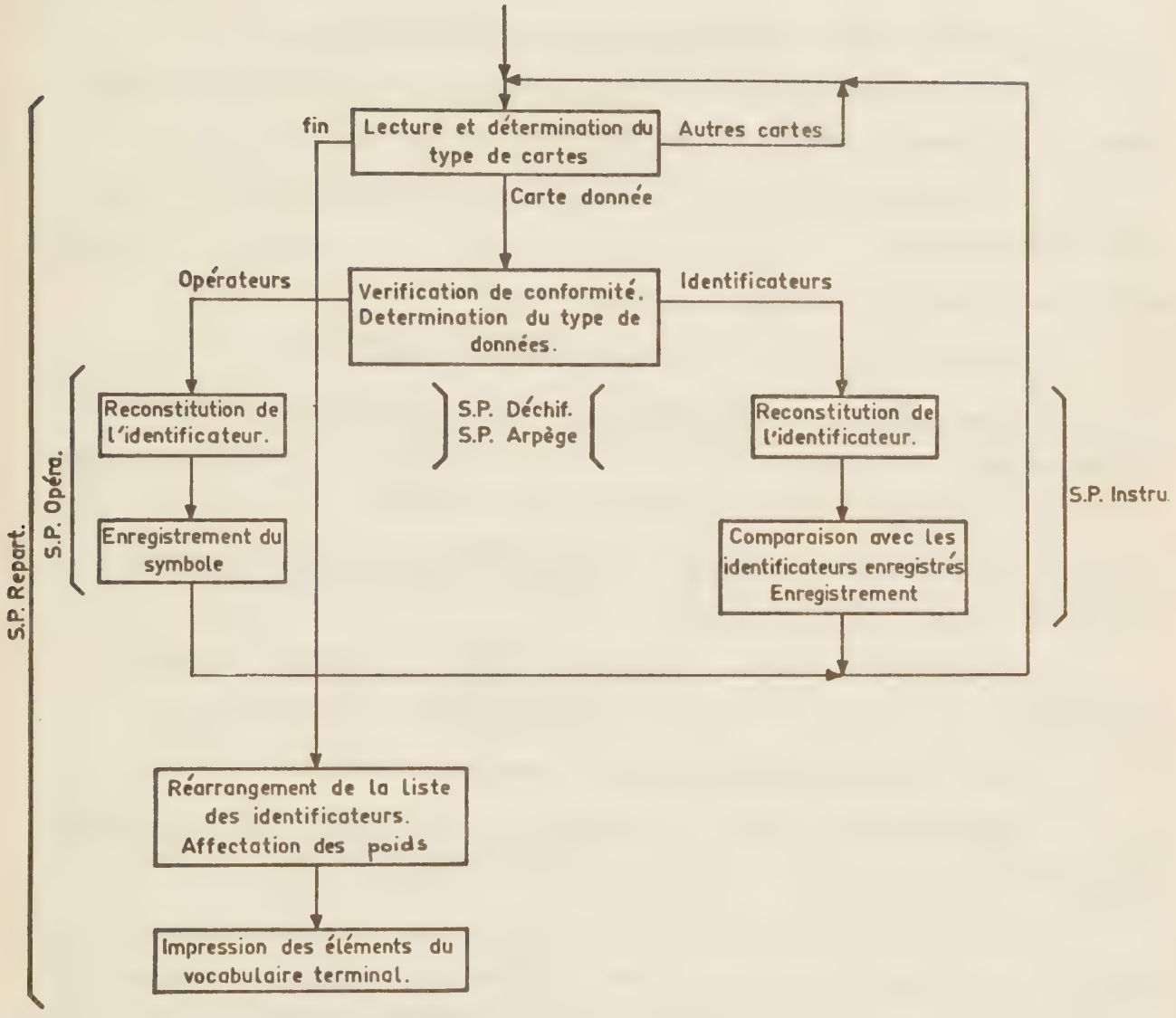
Le retour vers le sous-programme REPART s'effectue alors.

5.3 MODIFICATION DU VOCABULAIRE TERMINAL

Après le traitement d'une expression, l'utilisateur peut envisager de poursuivre le traitement, mais avec un vocabulaire terminal différent, en plaçant les cartes convenables (* en première colonne, Variables ou Opérateurs en zone "texte").

Le programme principal, assisté de PRELUD, détecte l'existence d'une carte de modification. Il en détermine la catégorie et affecte un poids convenable au mot PRGRAM. Il provoque alors le traitement exposé plus haut (cf.5.2). Le sous-programme, REPART, est chargé de vérifier que les listes présentées correspondent aux modifications envisagées.

PROGRAMME CHOPIN
CHAPITRE 5
CONSTITUTION DU VOCABULAIRE TERMINAL



ORGANIGRAMME FONCTIONNEL

FIG.5_1

- CHAPITRE 6 -

ANALYSE DE RECONNAISSANCE

La spécification a posteriori de la complémentation,

- le nombre, théoriquement illimité, de cartes utilisées pour l'impression d'une même expression (espacements non significatifs),

- l'obligation de vérifier la construction correcte de cette dernière avant de la réduire,

rendent cette analyse préalable nécessaire.

Nous avons exposé par ailleurs :

- les raisons qui conduisaient à préférer un langage à précédence d'opérateurs;

- les caractéristiques des deux grammaires CHOPIN.

Nous allons décrire l'organisation de l'analyse de reconnaissance, le vocabulaire terminal étant spécifié.

Dans un premier temps, nous caractériserons la grammaire "machine", c'est-à-dire la forme que présente cette grammaire pour l'ordinateur.

Dans la seconde période, nous décrirons cette analyse.

Nous donnerons ensuite un aperçu de la forme sous laquelle l'expression est représentée en machine.

Enfin, pour terminer ce chapitre nous indiquerons l'organisation fonctionnelle de cette analyse de reconnaissance, ainsi que les différents sous-programmes qui interviennent. La description détaillée de ceux-ci est faite dans la troisième partie du document.

6.1 GRAMMAIRE "MACHINE"

Les grammaires première et seconde formes ne diffèrent en fait que par le mode de développement des calculs. A savoir l'attribution à une seule des deux lois de composition de la propriété de distributivité, autrement dit de la priorité de provoquer le développement.

En différenciant les deux lois de composition non par leur nature mais par leur ordre de priorité (cf. Annexes 7 et 8) :

- loi de composition primaire : Θ ,
- loi de composition secondaire : $*$,

les deux grammaires sont ramenées à un seul et même type.

6.11 Vocabulaire non terminal

Le vocabulaire non terminal est modifié. Nous choisirons des termes qui soient compatibles avec ceux de la structure du treillis booléen normal :

<fonction>
<primaire>
<racine>
<radical>
<expression>
<représentation normale>

6.12 Ensemble des productions

Corrélativement, l'ensemble des productions devient :

<fonction> : : = λ_2 / λ_3 / <fonction>'
<primaire> : : = λ_4 / λ_1 / (<radical>) / <primaire>'
<racine> : : = <primaire> / <racine> Θ <primaire>
<radical> : : = <racine> / <radical> $*$ <racine>
<expression> : : = <fonction> := <radical>
<représentation normale> : : = $\dashv$ <expression> $\vdash$

6.13 Table des précédences conditionnées

La structure de grammaire à précédence d'opérateurs est maintenue. Les relations de précédence dans V_T sont conservées. A fortiori, le tableau des précédences, le tableau des précédences conditionnées gardent leurs formes, seuls changent les symboles des éléments terminaux.

Les figures 6.1 et 6.2 reproduisent l'ensemble V_N et le tableau des précédences conditionnées y afférents.

Il est possible de diminuer les dimensions de ce tableau. Les éléments terminaux : $\dashv$, $\vdash$ sont affectés implicitement. Ils ne peuvent être source d'erreur. D'autre part :

- $\dashv$ n'est jamais élément droite dans un graphe de relation,
- $\vdash$ n'est jamais élément gauche,

les colonnes et lignes correspondantes du tableau peuvent être supprimées. La partie utile du tableau se ramène à sept lignes et sept colonnes.

La figure 6.3 reproduit ce tableau ainsi que le tableau TABPRC qui en est l'image machine. L'affectation des valeurs numériques aux divers types de relation est semi-arbitraire. Son choix a été guidé par l'existence de propriétés communes au niveau de la signification ($\lessdot$, $\leqslant$, etc.....).

La figure 6.3c reproduit cette affectation.

6.14 Vocabulaire terminal

Du fait de l'ordinateur, le vocabulaire terminal doit subir un transcodage. D'ordinaire, celui-ci n'est qu'une simple bijection de V_T vers l'ensemble des symboles machine (en fait l'ensemble des nombres binaires naturels). Il en sera ainsi pour les identificateurs et les symboles autres que ceux afférents aux séparateurs de membres et aux deux lois de composition. Pour ces derniers, le transcodage sera double.

6.141 Symboles opératoires. Ponctuation

6.141.1 Séparateurs de membres

Bien que de natures différentes, les trois symboles d'affectation ou séparateurs de membres ne jouent qu'un seul et même rôle pour le traitement d'une expression. Nous ne leur avons fait correspondre qu'un seul et même élément terminal " :== ". Cependant ces symboles doivent être restitués sous la forme désirée lors de l'impression de sortie.

Ils subiront donc un premier transcodage, puis un codage d'affectation.

6.141.2 Lois de composition

Les symboles correspondants subissent eux aussi un double transcodage. Le premier qui consiste en l'affectation du symbole du type (ET, OU) est suivi d'un second lié à leur priorité, au type de forme normale choisie.

Ce type peut différer d'une expression à une autre. Il est caractérisé par (cf. Annexe 10) :

- l'absence de toute indication dans la zone réservée à cet effet pour une première forme;

- la présence d'un texte alpha-numérique dans cette même zone pour une seconde forme.

Selon le cas, le paramètre ISYMBO prendra la valeur 1 ou 2 respectivement.

La valeur de ISYMBO caractérise donc l'une ou l'autre des deux bijections de l'ensemble $\{ET, OU\}$ vers l'ensemble $\{\ominus, *\}$.

6.141.3 Tableaux de transcodage

Ils sont au nombre de deux.

Le premier OPERAT assure l'affectation du symbole à la nature de l'opérateur. Elle s'effectue au cours du traitement : "constitution du vocabulaire terminal".

Le second CLEDUT qui est un tableau à deux colonnes assure le transcodage

de la nature de l'opérateur à son rôle machine, selon la valeur de ISYMBO. Le transcodeur est fixé à demeure. Le transcodage a lieu au cours de l'analyse de reconnaissance (v. plus loin).

La figure 6.4 représente les correspondances nature-affectation, nature-rôle et schématise ces transcodages.

6.142 Identificateurs de variables

Nous avons vu (Chapitre 5) qu'en fin de phase "constitution du vocabulaire terminal" la liste des identificateurs était ordonnée. Une valeur, puissance entière de deux, afférente à son rang dans cette liste était allouée à chaque identificateur. Lors de l'analyse de reconnaissance, l'identificateur est promu primaire (ou variable booléenne au cours de la représentation des expressions booléennes). Cette promotion se traduit par un codage machine que nous allons considérer maintenant.

Les identificateurs sont les éléments générateurs du treillis booléen normal (cf. Annexes 6 à 8). Si n est leur nombre, ils vont créer 2n variables booléennes :

- n variables, en forme normale, qui correspondent aux identificateurs;
- n variables, en forme complémentée, qui correspondent aux identificateurs marqués du symbole de complémentation.

En choisissant une représentation ensembliste du treillis, les variables booléennes sont éléments d'ensembles. Ces derniers sont déterminés en extension par leur liste. Une représentation de celle-ci, de dimension réduite, est envisageable si chaque élément est codé en position (cf. convention d'écriture des nombres).

Le traitement ultérieur des représentations est facilité lorsque le codage de position est fonction de la nature de l'identificateur (entrée, secondaire, secondaire-sortie, sortie).

La réduction des expressions, leur transcodage littéral pour impression sont plus aisés si forme normale et complémentée d'une même variable sont rapidement discernables.

Enfin, l'indifférence d'une variable (sa non-existence dans l'ensemble) doit être décelable.

Ces différentes contraintes conduisent d'une part au rangement des identificateurs selon leur nature, d'autre part à affecter deux bits par identificateur de variable.

Deux classes de codage sont envisageables (v. figure 6.5) :

- chacune des deux formes d'une même variable est représentée par un bit. A chaque variable correspond un mot de 2n bits divisé en deux zones. Une des zones est réservée à la variable en forme normale, l'autre à la variable en forme complémentée.

- chaque variable y_i est représentée par un couple de mots (d_{y_i}, e_{y_i}) :
  - d_{y_i} appelé mot de disponibilité caractérise l'existence ou non de l'identificateur de rang ;
  - e_{y_i} appelé mot d'état caractérise la forme de cet identificateur pour la variable considérée.

Nous avons opté pour un codage de ce type :

- l'identificateur est aisément décelable. Il suffit de considérer un seul mot et non deux.

- la forme de l'identificateur est immédiatement reconnue;

- la recherche des variables communes et de leur forme, la mise en évidence de la relation d'ordre et de variables biformes sont facilitées.

Les conventions sont les suivantes (v. figure 6.6)

- identificateur de rang i ($1 \leq i \leq n$) : $i^{ème}$ bit de chaque mot (2^{i-1});

- identificateur x_i employé : $d_{x_i} = 2^{i-1}$, non employé : $d_{x_i} = 0$;

- identificateur x_i en forme normale : $e_{x_i} = 2^{i-1}$, en forme complémentée ou non employé : $e_{x_i} = 0$.

Le rang de l'identificateur est alloué au cours de la constitution du vocabulaire terminal. La promotion au rang de primaire a lieu au cours de l'analyse de reconnaissance en affectant à chaque variable de l'expression littérale un couple de mots dans la liste représentation de l'expression en machine (v. 6.3).

6.2 DESCRIPTION DE L'ANALYSE DE RECONNAISSANCE

Pour une grammaire à précédence d'opérateurs, l'analyse de reconnaissance est organisée eu égard aux propriétés des relations de précédence. La signification physique de celles-ci permet de mieux comprendre cette organisation. L'analyse de reconnaissance est faite lors de l'exploration directe de l'expression. Nous avons représenté, sur un tableau (figure 6.7), le sens que prennent les relations dans l'ensemble des éléments terminaux diminué des identificateurs de variables.

Nous voyons que cette analyse va consister essentiellement en la recherche de la borne droite d'une proposition, phrase ou locution (recherche aval). Celle-ci atteinte, la borne gauche est recherchée (recherche amont). Lorsqu'elle est atteinte, la recherche reprend pour la proposition, phrase ou locution qui suit. Elle se répète jusqu'à l'obtention de l'axiome, c'est-à-dire la représentation normale. Toute discordance est une erreur. La figure 6.8 schématise cette analyse de reconnaissance.

6.21 Eléments terminaux comparables

Alors que le nombre de symboles opératoires est petit (ils forment un ensemble de neuf éléments), celui des identificateurs de variables, possible, est énorme. Les fonctions des premiers sont imposés, il n'en est pas de même pour les secondes. Le nombre de caractères des premiers est limité à un, pour les seconds il varie de un à quatre. Il est donc plus facile de reconnaître les symboles opératoires que les identificateurs de variables. Aussi, est-il préférable de rechercher dans une expression les symboles opératoires et de mettre en évidence les relations de précédence qui existent entre eux. Cette façon de procéder permet, d'autre part, de rendre non significatifs les blancs ou espacements et de ce fait de permettre "d'aérer" les représentations en entrée.

6.22 Recherche aval

La recherche aval est celle qui s'effectue conjointement avec la lecture des caractères de la carte. Elle se poursuit aussi longtemps qu'une relation $\lessdot$, $\doteq$, ou $\gtrdot$ est mise en évidence.

Elle a pour but essentiel de reconnaître la limite droite d'une proposition, phrase ou locution.

Au début de l'analyse d'une nouvelle expression, l'élément terminal est introduit comme premier symbole opératoire de la "représentation machine" de l'expression.

Les colonnes de la carte sont alors lues les unes après les autres jusqu'à détection d'un symbole opératoire o_p. Les caractères autres que blancs qui le précèdent sont enregistrés pour constituer un <identificateur> . La relation de précédence qui existe entre o_p et l'élément terminal comparable, dernier enregistré. est reconnue par référence au tableau des précédences conditionnées.

6.221 Relation $\lessdot$.

Après vérification d'absence d'identificateur entre les deux éléments terminaux comparables, le symbole opératoire est enregistré dans le tableau SIGLOG et dans le tableau des pointeurs POINTS. Ce dernier permet d'assurer la gestion des éléments terminaux qui peuvent être bornes gauches.

La lecture se poursuit alors.

6.222 Relations $\lessdot$, $\leqslant$.

La présence, la nature et la conformité de l'identificateur de variables sont vérifiées. L'identificateur est enregistré après codage (v. 6.142) dans le tableau EQUATI. Le symbole opératoire est alors enregistré dans SIGLOG, pointé dans POINTS et la lecture reprise.

6.23 Recherche amont

La recherche amont se produit dès qu'une relation $\gtrdot$, ou $\geqslant$ a été mise en évidence par une recherche aval.

Elle va consister en la recherche de la borne gauche de la proposition, locution ou phrase dont la borne droite est le symbole opératoire o_p lu mais non enregistré, ni pointé. Elle pourra éventuellement mettre en évidence les bornes gauches des propositions ou locutions, membres de la phrase ou de la locution envisagée.

Pour ce faire, la relation qui existe entre les deux derniers éléments terminaux pointés est reconnue. Elle ne peut être que $\lessdot$, $\lessdot$, $\lessdot$ ou $\doteq$. Dans ce dernier cas, les deux derniers éléments terminaux pointés sont éléments de la phrase. Ils n'interviendront plus dans l'analyse. Ils sont supprimés du tableau POINTS par remontée de la pile. Dans les autres cas, seul le dernier élément terminal pointé est élément de la proposition, locution ou phrase. Il est supprimé de POINTS.

La relation qui existe entre o_p et le dernier élément de POINTS est alors considérée. Selon le cas, le déroulement de la rechercher diffère :

- $\doteq$: l'analyse de reconnaissance est terminée. La "représentation machine" de l'expression est constituée par les éléments des tableaux EQUATI et SIGLOG.

- $\gtrdot$, $\gtrdot$: la recherche amont reprend;

- dans les autres cas, o_p est enregistré dans POINTS, par descente de pile, ainsi que dans SIGLOG et la recherche aval est reprise.

6.3 REPRESENTATION MACHINE D'UNE EXPRESSION EN FORME BRUTE

L'expression, fournie en entrée, n'est soumise à aucune réduction logique au cours de l'analyse de reconnaissance. Elle est simplement transcodée. L'analyse de réduction se fera selon le même type de grammaire (à précédence conditionnée d'opérateurs). Il suffira donc de rechercher les relations entre éléments terminaux, autres que les identificateurs de variables. Sans cependant oublier que les réductions et opérations porteront sur ces derniers.

La représentation machine doit être telle que :

- les terminaux comparables soient aisément reconnus,

- les terminaux, identificateurs de variables, soient codés pour le traitement ultérieur et leur allocation aux différents opérateurs soit connue.

L'analyse de réduction va porter essentiellement sur le second membre (élément non terminal : <radical>). Le premier membre ne suit que l'affectation d'identité de l'expression. Cette différence doit apparaître dans la représentation.

6.31 Tableaux de la représentation

Deux tableaux sont utilisés :

- l'un SIGLOG (SIGnes LOGiques). Les terminaux comparables, $\dashv$ et $\vdash$ inclus, y sont enregistrés dans l'ordre de leur apparition dans l'expression.

- l'autre EQUATI (EQUATIon). Il contient la liste des identificateurs codés avec indication de l'adresse dans SIGLOG du premier symbole opératoire qui suit l'identificateur dans l'expression en entrée. Un second indicateur, utile pour l'analyse de réduction, est porté pour chaque identificateur. Sa valeur : zéro, un, caractérise la valeur syntaxique prise pour l'identificateur : racine ou primaire, radical.

6.32 Premier membre

Celui-ci est constitué par l'identificateur de la variable (sortie ou secondaire) sous forme normale ou complémentée. Le mot d'état ne peut être que zéro ou un, selon la parité du nombre de symboles de complémentation qui suivent l'identificateur dans l'expression littérale.

En représentation machine le symbole qui suit est toujours $:=$. La valeur du pointeur d'adresse est donc implicite et vaut deux. Les deux premiers éléments du tableau SIGLOG sont $\dashv$ et $:=$.

L'identificateur ne peut prendre que <fonction> pour valeur syntaxique. Cependant, le séparateur de membres choisi par l'utilisateur ne doit pas être perdu pour l'impression ultérieure. Il est enregistré au lieu et place de l'indicateur de valeur syntaxique.

6.33 Second membre

La représentation du second membre est constituée par :

- les images codées des formes normales des identificateurs de variables. Ces images sont placées entre la seconde ligne et la NUMTRE$^{\text{ième}}$ (NUMero de TeRmE) ligne du tableau EQUATI, dans la première et la seconde colonnes.

- les pointeurs d'adresse de l'élément terminal comparable qui suit l'identificateur. Ces pointeurs sont enregistrés en troisième colonne du tableau EQUATI.

- les images codées des éléments terminaux comparables. Elles sont enregistrées dans le tableau SIGLOG, entre la troisième et la NUMSIG^{ième} ligne (NUMéro de SIGne). Cette dernière caractérise l'élément $\vdash$.

6.34 Exemple de représentation

Nous avons reproduit figures 6.9 et 6.10 deux exemples de représentation machine.

La liste des identificateurs avec leur type,
l'affectation des poids,
l'expression littérale,
la correspondance entre les éléments et ceux des tableaux EQUATI et SIGLOG,
sont portées sur les figures.

6.4 ORGANISATION FONCTIONNELLE

L'analyse de reconnaissance proprement dite est effectuée sous le contrôle du sous-programme MVMNTS. Elle est déclenchée à partir du sous-programme TRADUC. Celui-ci est le premier sous-programme appelé après que le vocabulaire non-terminal ait été constitué (v. figure 6.11).

6.41 Sous-programme TRADUC.

Le sous-programme TRADUC organise l'analyse de reconnaissance et de réduction d'une expression. Nous bornerons ici son rôle à celui qu'il joue dans l'analyse de reconnaissance (v. figure 6.11).

Il est chargé de l'introduction implicite du symbole $\dashv$ et du symbole $\vdash$.

Par appel du sous-programme PRELUD, il provoque :

- la lecture des cartes de données l'une après l'autre,
- la détermination de leur type (première carte "données", carte "suite", carte "fin" etc....).

Il détermine donc les limites des expressions.

Il déclenche l'analyse de reconnaissance de l'expression carte par carte en appelant le sous-programme MVMNTS.

En fin d'analyse de reconnaissance, il assurera la poursuite convenable du traitement :

- analyse de réduction;
- analyse de l'expression suivante en cas d'erreur;
- retour au programme appelant.

6.42 Sous-programme MVMNTS

Le sous-programme MVMNTS a la charge de l'analyse de reconnaissance (cf. figure 6.12).

Par appel des sous-programmes :

- DECHIF : les caractères valides sont recherchés;
- SOLFEG : les éléments terminaux comparables sont reconnus;
- COPIER (entrée secondaire de DECHIF) : les identificateurs sont reconstitués;
- ACCORD : les identificateurs sont reconnus, codés et leur type caractérisé.

Les autres opérations de l'analyse :

- reconnaissance du type de relation de précédence;
- enregistrement des éléments terminaux dans les tableaux EQUATI et SIGLOG;
- gestion des pointeurs pour les recherches aval et amont;
- validation des types d'identificateurs;

sont du ressort de MVMNTS.

6.43 Description fonctionnelle

Le sous-programme PRELUD procède à la lecture de la carte (v. figure 6.13). Il indique au sous-programme EXECUT le type de celle-ci.

Lorsque la carte est du type "carte données", le sous-programme MVMNTS est appelé. Ce dernier recherche avec l'aide du sous-programme DECHIF le premier caractère significatif. La recherche terminée, le sous-programme SOLFEJ détermine

la nature du symbole : opérateur ou non. Dans ce dernier cas, l'appel de DECHIF par son entrée COPIER, puis celui du sous-programme ARPEGE provoquent la concaténation du caractère avec ceux préalablement reconnus, en vue de la formation de l'identificateur de variable.

A la fin de la concaténation, DECHIF, SOLFEJ sont successivement appelés jusqu'au moment où SOLFEJ reconnaît un symbole opératoire (ou de ponctuation).

MVMNTS met en évidence la relation qui existe avec le précédent symbole. Selon celle-ci, il :

- provoque l'enregistrement du symbole opératoire. Il appelle éventuellement le sous-programme ACCORD (reconnaissance, codage d'identificateur) et reprend la recherche aval;
- entraîne la recherche amont;
- marque la fin d'analyse.

En cas d'erreur l'analyse cesse, l'expression suivante est recherchée aux fins d'analyse. Cette recherche incombe à TRADUC et PRELUD.

PROGRAMME CHOPIN
CHAPITRE 6
ANALYSE DE RECONNAISSANCE

<Fonction> ::= λ_2 / λ_3 / <Fonction>'
<Primaire> ::= λ_1 / λ_2 / (<Radical>) / <Primaire>'
<Racine> ::= <Primaire> / <Racine> ⊖ <Primaire>
<Radical> ::= <Racine> / <Radical> * <Racine>
<Expression> ::= <Fonction> := <Racine>
<Représentation normale> ::= ⊣ <Expression> ⊢

Ensemble unifié des productions
FIG. 6.1

	⊣	:=	(	)	⊖	*	,	⊢
⊣	∠						∠	≐
:=		∢		▽	▽	▽	▽	▽
(		∢	≒	▽	▽	▽		
)			▷	▷	▷	▷	▷	▷
⊖		∢	▽	▽	▽	▽	▽	▽
*		∢	▽	▽	▽	▽	▽	▽
,		▷		▷	▷	▷	▷	▷
⊢								

Tableau unifié des précédences conditionnées
FIG. 6.2

	:=	(	)	,	⊖	*	⊢
⊣	∠		∠			≐	
:=		∢		▽	▽	▽	▽
(		∢	≒	▽	▽	▽	
)			▷	▷	▷	▷	▷
,	▷		▷	▷	▷	▷	▷
⊖		∠	▽	▽	▽	▽	▽
*		∢	▽	▽	▽	▽	▽

Représentation symbolique
-a-

	:=	(	)	,	⊖	*	⊢
⊣	7	8	8	7	8	8	2
:=	9	1	11	4	4	4	6
(	10	1	5	4	4	4	11
)	10	12	3	3	3	3	3
,	3	12	3	3	3	3	3
⊖	10	1	6	4	6	6	6
*	10	1	6	4	4	6	6

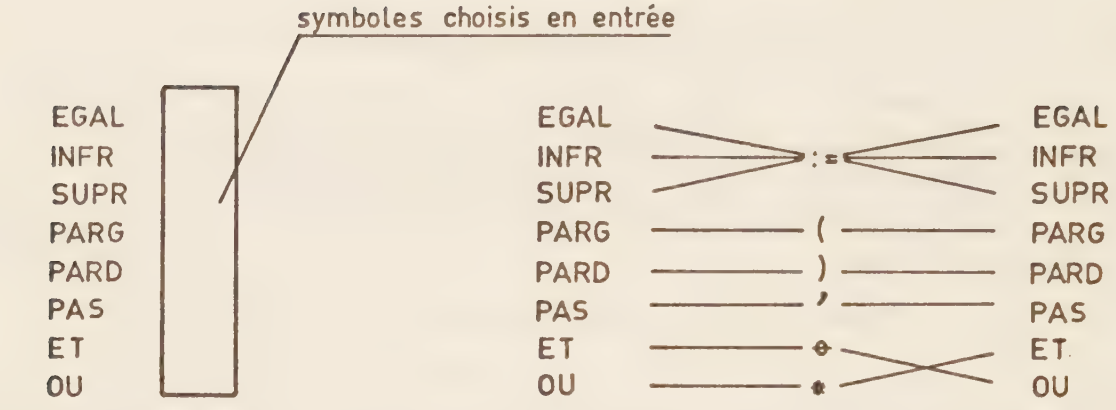
Représentation machine
-b-

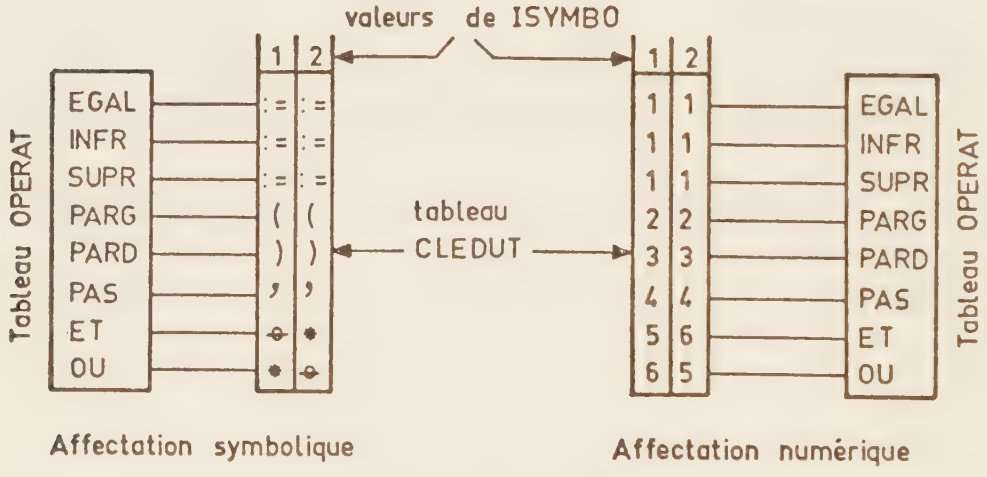
∢	1
≐	2
▷	3
▽	4
≒	5
▽	6
∠	7
erreur	≥ 8

Codage des symboles
-c-

TABLEAU DES PRECEDENCES TABPRC
FIG. 6.3

PROGRAMME CHOPIN
CHAPITRE 6
ANALYSE DE RECONNAISSANCE



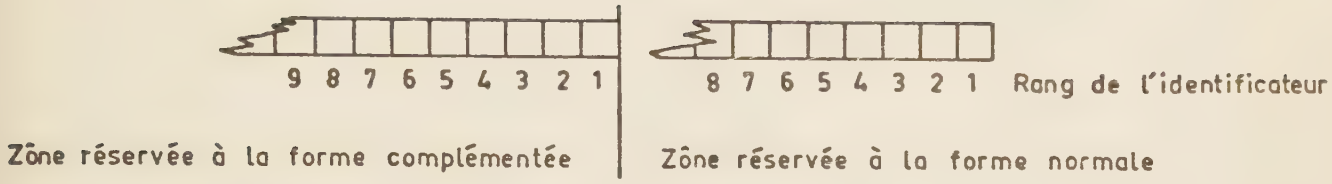


Correspondance entre les tableaux
OPERAT et CLEDUT

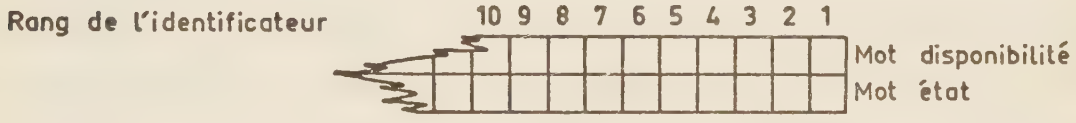
LE CODAGE MACHINE DES SYMBOLES OPERATOIRES

FIG. 6_4

PROGRAMME CHOPIN
CHAPITRE 6
ANALYSE DE RECONNAISSANCE

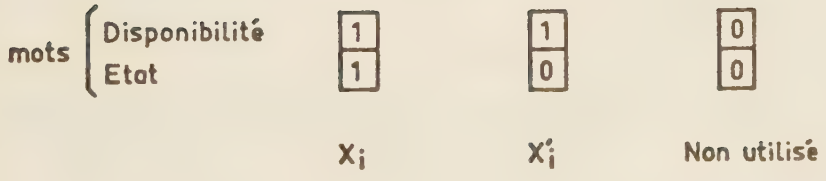


Représentation eu égard à la forme de la variable



Représentation eu égard aux caractéristiques de la variable

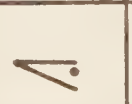
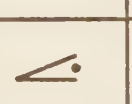
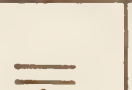
FIG.6_5



Valeurs des bits de rang i, affectés à l'identificateur x_i, dans les différents cas de représentation

FIG.6_6

PROGRAMME CHOPIN
CHAPITRE 6
ANALYSE DE RECONNAISSANCE

Type de la relation	Nature de l'identificateur intercalé	Opération à effectuer	Observations
	Néant	Poursuite de la lecture	
	Entrée, secondaire, secondaire-sortie		
	Secondaire, secondaire-sortie, sortie		
	Néant	Remonter vers la gauche	La limite droite de la proposition est atteinte La limite gauche est à rechercher
	Entrée, secondaire, secondaire-sortie		
	Entrée, secondaire, secondaire-sortie	Rechercher la borne gauche ou droite de la phrase proposition ou locution.	Une proposition, une locution, une phrase, est délimitée
	FIN de L'ANALYSE		

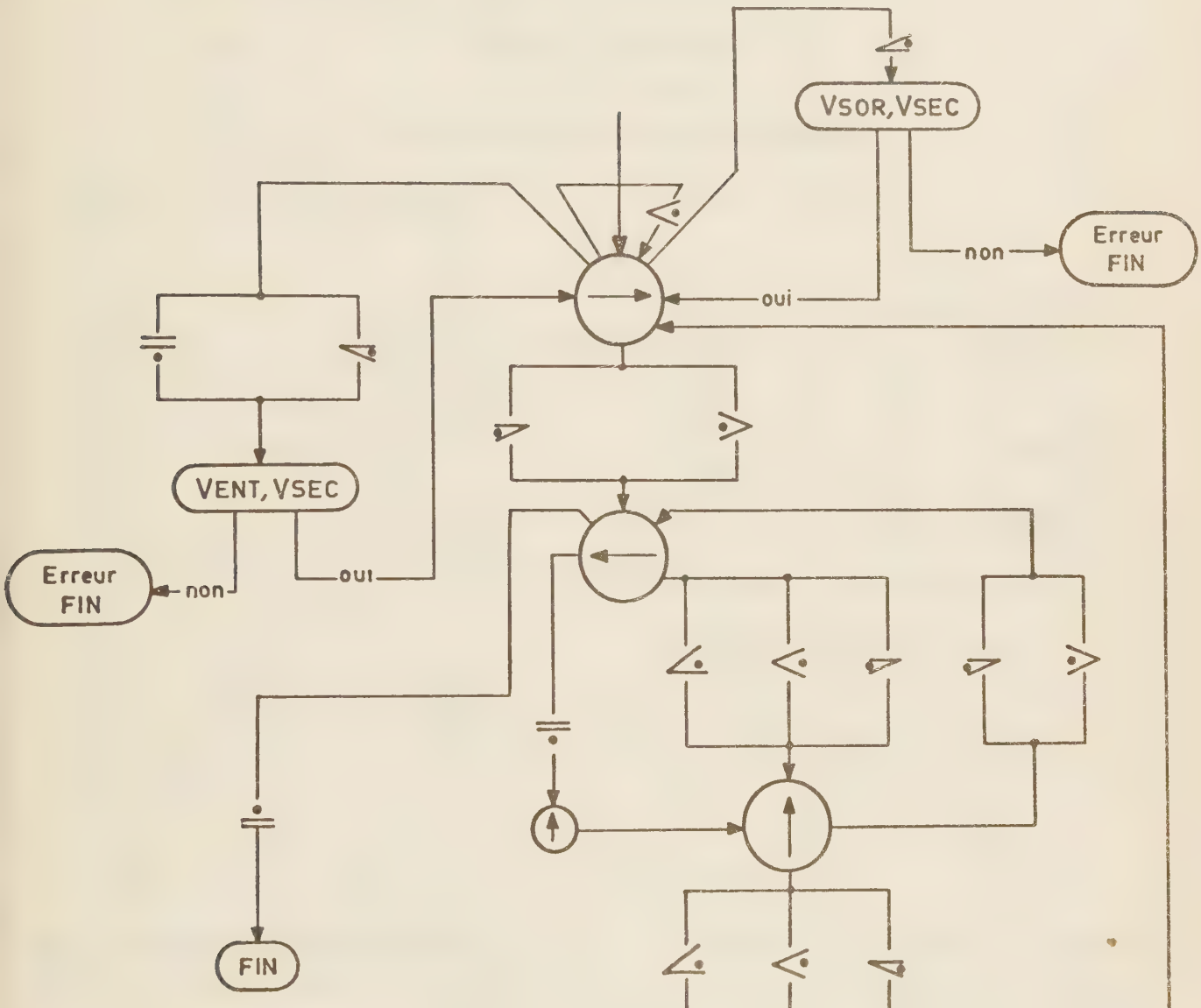
SIGNIFICATION PHYSIQUE DES RELATIONS DE PRECEDENCE CONDITIONNEE

FIG_6_7

PROGRAMME CHOPIN

CHAPITRE 6

ANALYSE DE RECONNAISSANCE



V....vérification de conformité de la nature de l'identificateur

→ recherche aval

← recherche amont

↑ remontée de la liste des terminaux comparables

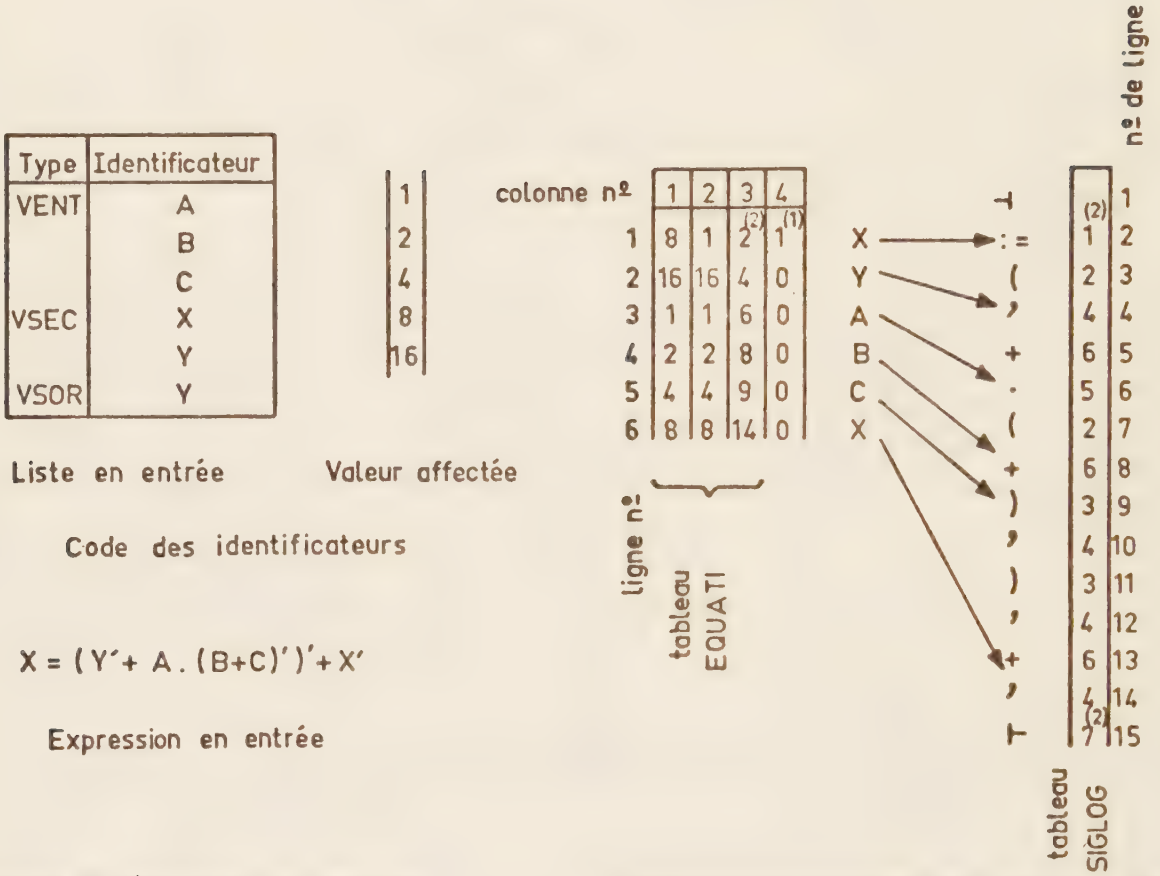
FIG.6_8

SCHEMA DE L'ANALYSE DE RECONNAISSANCE

PROGRAMME CHOPIN

CHAPITRE 6

ANALYSE DE RECONNAISSANCE



(1) Adresse du séparateur de membre dans le tableau OPERAT

(2) Affectation implicite

Représentation machine de l'expression avec indication des images littérales et de leur situation

1er EXEMPLE DE REPRESENTATION MACHINE

FIG. 6_9

PROGRAMME CHOPIN
CHAPITRE 6
ANALYSE DE RECONNAISSANCE

= Liste des variables=

Entrée	Secondaire	Secondaire. Sortie	Sortie
INIT X0 X2	Y_1	RAZ	Q_1

Variables	Rang	Valeur codée
INIT	1	1
X0	2	2
X2	3	4
RAZ	4	8
Y_1	5	16
Q_1	6	32

= Liste des opérateurs =

EGAL	=	1
INFR		
SUPR		
PARG	(	2
PARD	)	3
PAS	'	4
ET	.	5
OU	+	6

valeur allouée

$Q_1 = INIT.\ ((X_0' + X_2)' + (Y_1 + X_0 . X_2')) + X_0$

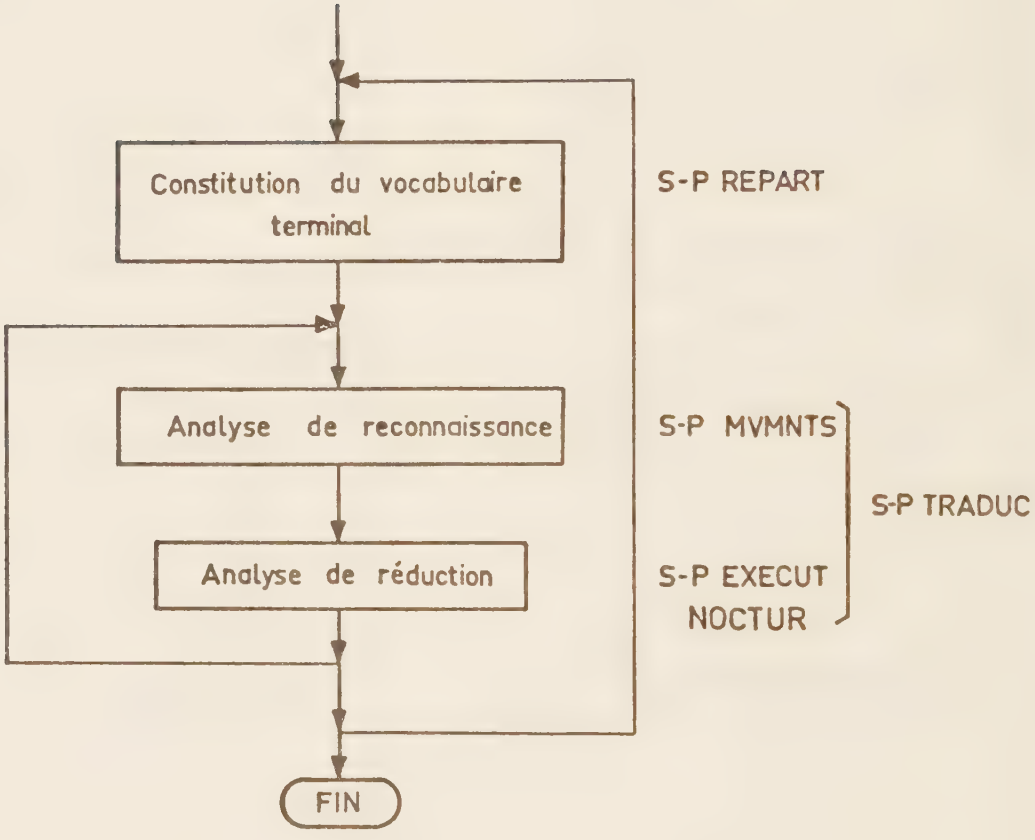
	EQUATI 1	2	3	4
1	32	32	2	=
2	1	1	3	0
3	2	2	6	0
4	4	4	8	0
5	16	16	12	0
6	2	2	13	0
7	4	4	14	0
8	2	2	18	0

SIGLOG		
0	1	⊣
1	2	=
5	3	.
2	4	(
2	5	(
4	6	'
6	7	+
3	8	)
4	9	'
6	10	+
2	11	(
6	12	+
5	13	.
4	14	'
3	15	)
3	16	)
6	17	+
7	18	⊢

2e EXEMPLE DE REPRESENTATION MACHINE

FIG. 6_10

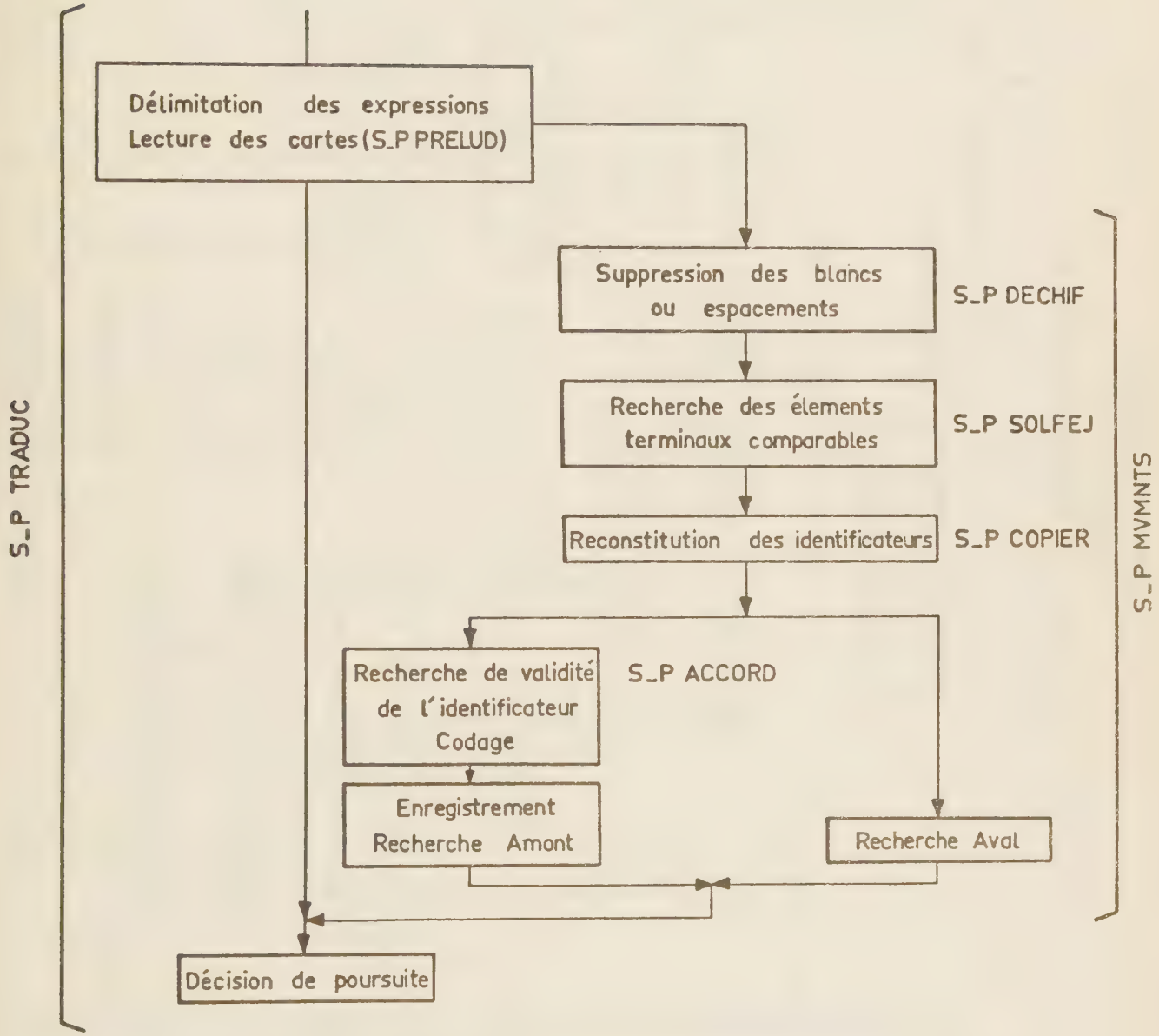
PROGRAMME CHOPIN
CHAPITRE 6
ANALYSE DE RECONNAISSANCE



ORGANISATION SEQUENTIELLE DE L'ANALYSE DE RECONNAISSANCE

FIG _ 6 _11

PROGRAMME CHOPIN
CHAPITRE 6
ANALYSE DE RECONNAISSANCE



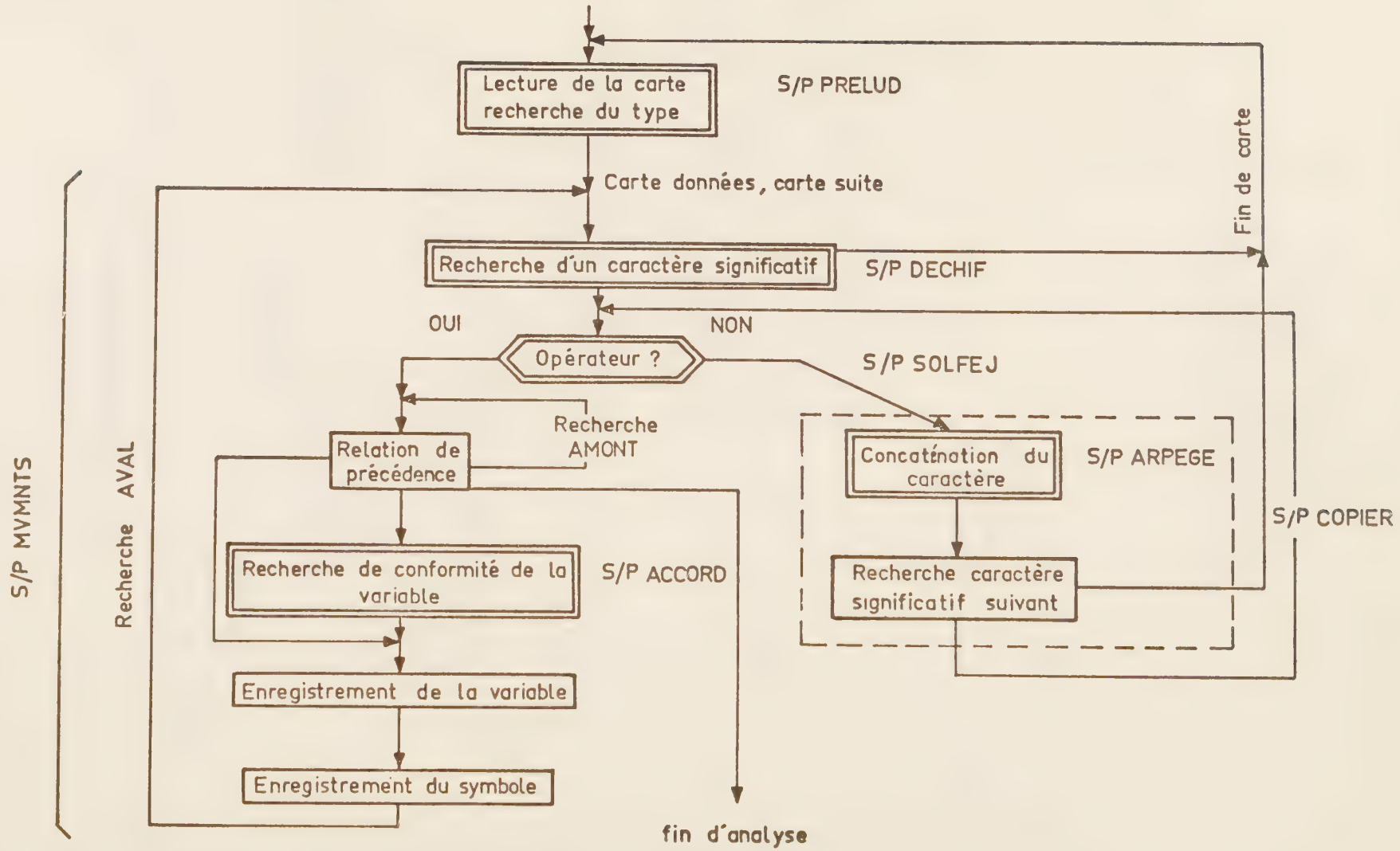
ORGANISATION FONCTIONNELLE

FIG_6_12

PROGRAMME CHOPIN

CHAPITRE 6

ANALYSE DE RECONNAISSANCE



SCHEMA DE L'ANALYSE

FIG. 6_13

- CHAPITRE 7 -

"REPRESENTATION MACHINE" DU TREILLIS BOOLEEN NORMAL

La "représentation machine" des chaînes terminales du langage CHOPIN influe tant sur l'encombrement mémoire que sur le temps de traitement. Les paramètres, afférents à ce dernier, sont essentiellement les organisations séquentielles des opérations qui assurent la composition des éléments du treillis, leur comparaison.

Après avoir considéré la représentation ensembliste d'un treillis booléen normal, nous caractériserons sa représentation machine.

7.1 REPRESENTATION ENSEMBLISTE

Nous avons vu (Annexe 7) que le traitement de représentations booléennes normales était envisageable en ne considérant que les ensembles associés aux racines et aux radicaux. Nous allons compléter la définition des correspondances entre ces éléments par celle des relations entre structures. Nous caractériserons alors un homomorphisme.

7.11 Ensembles en correspondance

Considérer

- un radical A_i de racines génératrices α_{ij}

$$A_i = \overset{n_i}{\underset{j=1}{*}} \alpha_{ij}$$

revient à envisager l'ensemble de ces racines génératrices;

- une racine α_{ij} de variables génératrices y_{ijk}

$$\alpha_{ij} = \overset{p_{ij}}{\underset{k=1}{\Theta}} y_{ijk}$$

revient à considérer l'ensemble de ces variables booléennes génératrices;

- une variable booléenne y_{ijk} est envisager un élément de l'ensemble générateur G, affecté ou non de la complémentation.

En remarquant que :

- une racine est un radical constitué par une seule racine;
- une variable booléenne est une racine de degré 1;

nous pouvons envisager une correspondance entre $\mathcal{B}$ et $\mathbb{E}$ caractérisée comme suit :

$$\mathbb{G} = \{y_j, j \in [1,2n] ; y_j = x'_i \quad ou \quad y_j = x_i , x_i \in \mathcal{G}\}$$

$$\mathcal{S}_{(\mathbb{G})} = \{\text{parties de } \mathbb{G}\}$$

$$\mathbb{E} = \{\text{parties de } \mathcal{S}_{(\mathbb{G})}\}$$

La correspondance est alors une relation :

$$\mathcal{B} \xrightarrow{\quad \mathcal{C} \quad} \mathbb{E}$$

telle que

$$\forall A_i = \overset{m_i}{\underset{j=1}{*}} \left(\overset{p_{ij}}{\underset{k=1}{\oplus}} y_{ijk} \right) \in \mathcal{B} :$$
$$\exists \mathcal{A}_i \in \mathbb{E} ; \mathcal{A}_i = \{\{y_{ijk}, k \in [1,p_{ij}] ; y_{ijk} \in \mathbb{G}\} ; j \in [1,m_i]\} \qquad \text{<A-71>}$$

Eu égard aux propriétés de $(\mathcal{B}_{(\mathcal{G})}, \oplus, *, ')$ et à la définition de cette relation :

" $\mathcal{C}$ est une application injective de $\mathcal{B}$ dans $\mathbb{E}$ " <T-71>

La propriété est évidente.

7.12 Homomorphisme défini par $\mathcal{C}$.

En transformant la définition en compréhension de $\mathcal{A}_i$, donnée dans l'énoncé de l'axiome 71, nous allons faire apparaître les opérations induites par l'application $\mathcal{C}$:

$$\mathcal{A}_i = \bigcup_{j=1}^{m_i} \left\{ \bigcup_{k=1}^{p_{ij}} \{y_{ijk}\} \right\}$$

Il est alors facile d'établir les correspondances, données ci-après, induites par $\mathcal{C}$.

$$\mathcal{C}_{(A_i * A_j)} = \mathcal{C}_{(A_i)} \cup \mathcal{C}_{(A_j)}$$

<A-72>

où $\cup$ est la réunion dans $\mathbb{E}$. Elle conserve ses propriétés.

$$A_i = \overset{m_i}{\underset{j=1}{*}} \left(\overset{p_{ij}}{\underset{h=1}{\ominus}} y_{ijh} \right) \qquad B_f = \overset{m_f}{\underset{g=1}{*}} \left(\overset{p_{fg}}{\underset{h=1}{\ominus}} z_{fgh} \right)$$

$$\mathcal{C}_{(A_i * B_f)} = \bigcup_{j=1}^{m_i} \left(\bigcup_{g=1}^{m_f} \left\{ \bigcup_{h=1}^{p_{ij}} \{ y_{ijh} \} \bigcup_{h=1}^{p_{fg}} \{ z_{fgh} \} \right\} \right)$$

$$\mathcal{C}_{(A_i * B_f)} = \mathcal{C}_{(A_i)} \ominus \mathcal{C}_{(B_f)} \qquad \text{<A-73>}$$

$\ominus$ sur $\mathbb{E}$ équivaut donc à la réunion dans $\mathbb{E}$ des réunions dans l'ensemble des parties de $\mathbb{G}$. Il est facile de voir que $\ominus$ est loi de composition dans $\mathbb{E}$, commutative, associative, distributive par rapport à $\cup$, d'élément neutre $\mathbb{E}$. <A-74>

$$\mathcal{C}_{(N)} = \{ \} = \emptyset \qquad \text{<A-75a>}$$

$$\mathcal{C}_{(Y)} = \mathbb{E} \qquad \text{<A-75b>}$$

Le théorème de MORGAN dans $\mathcal{B}$ induit un théorème homologue dans $\mathbb{E}$, que nous énoncerons (Théorème 72) :

"le complémentaire d'un élément $\mathcal{A}$ de $\mathbb{E}$ est un élément de $\mathbb{E}$, obtenu en remplaçant dans l'expression de $\mathcal{A}$:

- la loi de composition $\cup$ par $\ominus$,
- la loi de composition $\ominus$ par $\cup$,
- les variables en forme normale par leur forme complémentée et inversement,
- la partie vide par la partie pleine et vice-versa".

A la relation d'ordre $\dashv$ dans $\mathcal{B}$ correspond l'inclusion $\subset$ dans $\mathbb{E}$. <A-76>

Nous dirons qu'il y a eu contraposition lorsque $\mathcal{A}$ n'est pas une variable booléenne.

Théorème 73

Le sous-ensemble $\mathbb{R}$ de $\mathbb{E}$ dans lequel $\mathcal{C}$ prend ses valeurs est muni d'une structure de treillis booléen normal : ($\mathbb{R}$, $\ominus$, $\subset$, $'$).

Cette propriété se déduit des précédentes.

7.2 REPRESENTATION MACHINE

Nous avons indiqué plus haut (6.142) le mode de représentation d'une variable booléenne :

à toute variable booléenne y_j , engendrée par un identificateur de rang i ($1 \leqslant i \leqslant n$), est affecté un couple de mots $(d_{y_i}, e_{y_i}) = \mathcal{C}_n(y_i)$, tels que :

- $d_{y_i} = 2^{i-1}$
- $e_{y_i} = 0$, si y_i est sous forme complémentée,
- $e_{y_i} = d_{y_i}$, si y_i est sous forme normale.

En convenant de transposer en machine la représentation ensembliste, nous associons

- à toute racine (ou variable) : un couple de mots;
- à tout radical : la liste de ses racines génératrices;
- à la loi secondaire : la réunion des éléments des listes;
- à la composition primaire de deux racines : la racine engendrée par les variables communes et non communes, sous réserve de non nullité;
- à la composition primaire de deux radicaux : la réunion en une liste des compositions primaires, partielles des racines.

Nous allons terminer la caractérisation de cette représentation par celle des propriétés qui intéressent la réduction booléenne : relation d'ordre, racine nulle, racine unité, consensus, adjacence .... etc.

7.21 Relation d'ordre

Théorème 74

Une condition nécessaire pour que deux racines, α et β , représentées par les couples (d_α , e_α), (d_β , e_β) soient telles que

$$\alpha \prec \beta$$

est que

$$d_\alpha \geqslant d_\beta$$

<T-A75> , $\beta = \overset{n}{\underset{i=1}{\oplus}} y_i$: $\alpha \prec \beta \longrightarrow \alpha = \overset{n}{\underset{i=1}{\oplus}} y_i \ominus \alpha$.

$d_\beta = \sum_{i=1}^{n} 2^{i-1}$, $d_\alpha = q + \sum_{i=1}^{n} 2^{i-1}$: $d_\alpha \geqslant d_\beta$

C.Q.F.D.

Il en est de même de e_α et e_β.

Si Λ représente l'intersection bit à bit, V la réunion bit à bit, nous pouvons écrire :

$$d_\alpha \Lambda d_\beta = d_\beta \qquad e_\alpha \Lambda e_\beta = e_\beta$$

Inversement si nous envisageons $\mathcal{C}_{n(\alpha)} = (d_\alpha, e_\alpha)$, $\mathcal{C}_{n(\beta)} = (d_\beta, e_\beta)$ tels que :

$$d_\alpha \Lambda d_\beta = d_\beta \qquad e_\alpha \Lambda e_\beta = e_\beta$$

il est évident que $\alpha \prec \beta$

Théorème 75

La condition nécessaire et suffisante pour que deux racines α et β, d'images machine (d_α, e_α), (d_β, e_β) respectivement, soient telles que :

$$\alpha \prec \beta$$

est que

$$d_\alpha \Lambda d_\beta = d_\beta$$

et

$$e_\alpha \Lambda e_\beta = e_\beta$$

7.22 Racine nulle, racine unité

7.221 Racine unité

η racine unité (A7.351) a pour image machine la liste complète de toutes les racines que peut engendrer l'ensemble G. La dimension d'une telle liste est 2^n (n égale le nombre d'identificateurs).

En caractérisant η par une racine, nous ramenons cette dimension à un couple (d_η, e_η). Les valeurs de d_η et e_η, eu égard au théorème 75 et aux propriétés des treillis booléens (<T-A610>), doivent être les plus faibles possibles, c'est-à-dire 0 :

$$\mathcal{C}_{n(\eta)} = (0,0)$$

7.222 Racine nulle

N, racine nulle (A7.351) a pour image l'ensemble vide, c'est-à-dire une liste vide. Nous avons préféré malgré tout la représenter par un couple de mots (d_N , e_N). Eu égard à <T-A75> et <T-A611> , les valeurs de d_N et e_N doivent être les plus élevées possibles :

$$\mathcal{C}_{M(N)} = (2^n, 2^n)$$

7.223 Représentations machine de η et de N .

Les représentations précédentes offrent deux inconvénients :

- le critère de reconnaissance est double : 0 ou 2^n ;
- la complémentarité est difficilement mise en évidence.

Aussi avons-nous pris une représentation intermédiaire qui offre en plus l'avantage d'être mnémonique :

$$\mathcal{C}_{M(\eta)} = (0, q) \qquad 1 \leqslant q \leqslant 2^{n-1}$$

$$\mathcal{C}_{M(N)} = (0, 0)$$

Les racines nulle ou unité se reconnaissent immédiatement à la valeur de leur mot de disponibilité.

La racine nulle a un mot d'état de valeur nulle.

7.23 Composition primaire de deux racines

7.231 Composition nulle

Eu égard aux propriétés de la loi de composition primaire, de la racine nulle (cf. A7.3) :

La condition nécessaire et suffisante pour qu'une composition de deux racines soit racine nulle est qu'il existe, au moins, une variable biforme entre ces racines

<T-76>

7.232 Racines liées par ⊣

Par définition de la relation ⊣ (A7.33), si deux racines sont liées par ⊣ , leur composition est la racine postérieure.

7.233 Cas général

Dans le cas général, il est facile de voir que :

$$\mathcal{C}_{n(\alpha)} = (d_\alpha, e_\alpha),\ \mathcal{C}_{n(\beta)} = (d_\beta, \alpha_\beta) \quad :\quad \mathcal{C}_{n(\alpha \circ \beta)} = (d_\alpha \vee d_\beta,\ e_\alpha \vee e_\beta)$$

c'est-à-dire la racine engendrée par l'ensemble de toutes les variables génératrices, communes ou non.

7.24 Composition secondaire de deux racines

7.241 Consensus d'ordre 2

Théorème 77

La condition nécessaire et suffisante pour qu'un consensus, d'ordre 2, existe relativement à deux racines d'images machine (d_1, e_1), (d_2, e_2) est que :

$$d_1 \wedge d_2 \wedge [(e_1 \wedge \bar{e}_2) \vee (\bar{e}_1 \wedge e_2)]$$

soit une puissance entière de deux. L'exposant de la puissance caractérise la variable biforme.

$\bar{e}$ représente la complémentation bit à bit de e .

Cette propriété est évidente.

Corollaire 71

La condition suffisante pour que deux racines, d'images machine (a, b), (c, d), n'engendrent pas de consensus d'ordre 2 est que $a \wedge c$ soit nul.

7.242 Racines adjacentes

Eu égard à la définition des racines adjacentes (<C-A70>), nous pouvons énoncer :

Théorème 78

La condition nécessaire et suffisante pour que deux racines soient adjacentes est

- que leurs images admettent le même mot de disponibilité,
- qu'elles admettent un consensus d'ordre 2.

7.243 Composition unité

Théorème 79

Dans une représentation normale, la condition nécessaire et suffisante pour que deux racines engendrent par composition secondaire la racine unité est qu'elles soient de degré un et complémentaires.

En effet

$$\alpha * \beta = \eta \longrightarrow \alpha = \beta'$$

$$\forall \beta = \overset{p}{\underset{i=1}{\odot}} y_i \quad : \quad \beta' = \overset{p}{\underset{i=1}{*}} y_i$$

pour que β' soit une racine il faut que $p = 1$

Il en est de même pour α .

7.244 Composition secondaire de deux racines

La constitution du radical par enregistrement de la liste formée par les deux racines ne se fera qu'après vérification des cas précédents et, lors de l'existence d'un consensus, de la précédence des composants relativement au consensus.

- CHAPITRE 8 -

ANALYSE DE REDUCTION

L'expression à traiter est conforme, en présentation et en écriture, aux règles de la grammaire choisie. Sa représentation machine brute est obtenue. Le premier membre est ramené à la seule fonction, en forme normale ou complémentée. Les différentes compositions logiques qui portent sur le second membre doivent être maintenant exécutées, eu égard aux :

- réductions syntaxiques de la grammaire;
- réductions sémantiques, booléennes du langage.

Nous allons situer les premières relativement aux productions syntaxiques.

Nous décrirons ensuite, globalement, l'analyse de réduction. Nous y retrouvons l'analyse de reconnaissance avec un sens d'exploration inverse, entremêlée avec la réduction des représentations en forme allégée des compositions booléennes. L'analyse ne se borne pas à la simple reconnaissance syntaxique mais à la recherche des chaînes terminales. Elle prendra fin avec la recherche des représentations complète et première, dont nous détaillerons le principe.

Nous examinerons enfin l'organisation fonctionnelle de l'analyse et caractériserons les différents sous-programmes y afférents. Rappelons que les descriptions, plus détaillées de ces derniers, sont données en troisième partie.

8.1 ENSEMBLE DES REDUCTIONS

Normalement, les réductions sont les relations réciproques des productions dans γ_T (cf. Annexe I). L'utilisateur peut spécifier un premier membre d'expression par un identificateur de variable suivi d'un nombre quelconque de complémentations. Pour des raisons évidentes de réduction d'encombrement, ce premier membre est traité lors de l'analyse de reconnaissance. Il est donc réduit. L'analyse de réduction ne l'envisagera pas. Il caractérisera la fin de la réduction syntaxique.

L'ensemble des réductions syntaxiques à considérer se limite aux seules réductions relatives au traitement du second membre , les identificateurs ayant été

créés lors de l'analyse de reconnaissance :

<identificateur> / (<radical>) / <primaire>' ⟶ <primaire>
<primaire> / <racine> ⊙ <primaire> ⟶ <racine>
<racine> / <radical> * <racine> ⟶ <radical>
<fonction> :≡ <radical> ⟶ <expression>
⊣ <expression> ⊢ ⟶ <représentation normale>

Simultanément, les chaînes terminales obtenues sont représentées par leur forme allégée pour les trois premières réductions. Pour les deux dernières, les représentations sont les représentations complète et première.

8.2 DESCRIPTION DE L'ANALYSE DE REDUCTION

Pour sacrifier à l'habitude de complémenter a posteriori, le symbole de complémentation est écrit le dernier, c'est-à-dire à droite.

Pour éviter un trop grand nombre d'allées et venues lors de l'exploration (une gestion de piles, de listes trop compliquée), nous avons préféré faire celle-ci dans le sens inverse (cf. chapitre 1 : figures 1.2 et 1.3). Nous appliquons ainsi le théorème de MORGAN au cours de l'élaboration de la représentation et non sur la représentation élaborée.

Comme pour l'analyse de reconnaissance, et d'une façon plus prononcée, l'analyse de réduction est constituée de deux parties d'inégales importances :

- la recherche amont (en sens inverse) sur l'expression littérale par remontée de liste en machine, au cours de laquelle la borne gauche de la proposition, locution ou phrase est l'objectif;

- la recherche aval (dans le sens direct), effectuée par descente de pile, qui a pour but la reconnaissance de la borne droite et la reconstitution de la proposition, phrase ou locution. Elle s'accompagne de la "réduction sémantique".

La recherche débute avec le dernier élément de SIGLOG comme élément droite.

8.21 Recherche amont

La recherche amont (v. figure 8.1) est liée à la mise en évidence de l'une des deux relations de précédence conditionnée :

- $\geq$ qui caractérise deux éléments terminaux comparables, adjacents.
- $\not\geq$ qui caractérise deux éléments terminaux comparables, séparés par un élément syntaxique.

8.211 Eléments comparables adjacents

Parmi les conséquents droits, le symbole de complémentation se particularise. Il marque une contraposition (cf. 7.12 - <T-72>) par le théorème de MORGAN. Il est en effet accolé à un séparateur droit. Si le nombre de contrapositions rencontrées est pair, elles n'ont aucun effet. Par contre, s'il est impair, le théorème 72 doit être appliqué. La réduction sémantique n'interviendra qu'au cours de la recherche amont, la parité du nombre des contrapositions est caractérisée par le signe de la valeur algébrique du paramètre NEGATI (NEGATIon) :

- positif lorsque le nombre est pair,
- négatif lorsqu'il est impair.

Quel que soit le cas, le conséquent droit n'est pas élément de la locution, mais en est un délimiteur. Son adresse dans le tableau SIGLOG est enregistrée dans la pile POINTS (POINTeur Symboles). Cependant, lorsque les deux éléments terminaux sont des ', le conséquent droit n'a aucune raison de venir s'ajouter à la pile.

8.212 Eléments comparables non adjacents

L'identificateur compris entre les deux éléments comparables est promu <primaire> . La pile POINTE (POINTeur Elément), réservée à l'enregistrement de leur adresse dans EQUATI, est augmentée. Cette promotion s'accompagne d'une modification de la proposition représentée par le couple de mots (d,e) lorsque NEGATI est négatif :

- e prend la valeur d si sa valeur préalable était nulle,
- la valeur 0 si elle en était différente.

La pile POINTS s'accroît d'un élément.

Quelle que soit la relation $\geq$ ou $\not\geq$, la recherche reprend en considérant l'élément terminal qui précède dans le tableau SIGLOG.

8.22 Recherche aval

Elle se subdivise en deux recherches (v. figure 8.2).

8.221 Reconnaissance de la borne gauche

Elle est consécutive à la mise en évidence d'une relation $\lessdot$. En effet, la phrase la plus simple qu'il soit possible d'engendrer par la grammaire CHOPIN (cf. A1.32) dans un second membre est la proposition. Elle ne peut contenir qu'un identificateur.

Une réduction syntaxique va être effectuée. Elle sera suivie éventuellement d'une réduction sémantique dont le type dépend de la parité des contrapositions (signe de NEGATI).

8.221.1 Complémentation

Le seul cas où ' est antécédent droit est celui où il est à droite d'un identificateur (cf. table de précédences conditionnées : figure 6.2). Il s'agit d'une complémentation. L'identificateur est promu <primaire> avec mise à zéro du mot d'état si la valeur algébrique de NEGATI est négative.

8.221.2 Composition primaire ou secondaire

Ces compositions porteront progressivement sur

- des <identificateurs> promus <primaires> qui sont des éléments du tableau EQUATI;
- des compositions. Mais les résultats de ces compositions sont soit des racines, soit des radicaux. Encore que ces derniers puissent être monoracines.

Il nous faut donc différencier les opérandes par le nombre de leurs racines: une ou plusieurs. Il était tout à fait normal de conserver les premières dans le tableau EQUATI et d'enregistrer les secondes dans le tableau EXPRES (EXPRESsion), réservé aux représentations complètes. Mais l'exploration se fait en référence au tableau EQUATI, l'indication du type d'opérandes doit y être portée. Ce rôle est dévolu à la quatrième colonne du tableau. Enfin une pile POINTX (POINTeur eXpressions permet d'avoir les adresses des limites des opérandes multi-racines dans EXPRES.

Les réductions sémantiques, ou compositions des formes allégées, sont effectuées eu égard au signe de la valeur algébrique de NEGATI :

- négatif : les lois de composition sont permutées;
- positif : la loi de composition indiquée est appliquée.

Lorsque la réduction syntaxique et la réduction sémantique sont terminées :

- les piles POINTE et POINTS sont remontées d'une unité. Un opérande et un signe ont été traités.

- la pile POINTX a été mise à jour si la composition est un radical multiracine;

- la recherche de la borne droite de la phrase ou locution débute.

8.222 Recherche de la borne droite

Cette recherche diffère de la recherche amont par le fait qu'elle s'effectue par remontée des piles POINTS et POINTE. Les seuls types de relations qui peuvent être reconnus sont : $\leqslant$, $=$, $\geqslant$.

8.222.1 Relation $\leqslant$

Nous retrouvons le cas normal envisagé supra (8.221) avec les réductions syntaxique et sémantique.

8.222.2 Relation $=$

Seule une réduction syntaxique intervient. Elle se traduit par remontée de la pile POINTS pour supprimer l'indication du groupement. Elle s'accompagne d'une modification du signe de NEGATI si le délimiteur droit est précédé du symbole ' .

Elle est suivie d'une nouvelle recherche de la borne gauche, puisque les éléments terminaux comparables font partie de la locution et n'en sont pas les bornes.

La recherche aval se poursuit.

8.222.3 Relation $\geqslant$

La borne droite est atteinte. La pile POINTE est mise à jour. La recherche amont reprend.

8.23 Fin de l'analyse syntaxique de réduction

Elle intervient lorsque les éléments terminaux comparables sont $\dashv$ et $:=$, c'est-à-dire sur une relation du type $\lessdot$.

La représentation obtenue est sous forme allégée. Elle doit être transformée en forme complète, puis en une forme première.

8.3 REDUCTION DE LA FORME ALLEGEE

La mise en évidence de la relation $\lessdot$ (les éléments terminaux comparables sont alors $\dashv$ et $:=$) caractérise la fin de l'analyse syntaxique de réduction. La réduction sémantique a conduit à une représentation en forme allégée qui doit être réduite en sa représentation complète. Enfin, cette dernière est réorganisée pour mettre en évidence une représentation première.

8.31 Recherche de la représentation complète

8.311 Principes

8.311.1 Répartition en classes

Le nombre d'identificateurs (2 à 32) peut être élevé. Il en est de même de celui des racines. La méthode de recherche des consensus itératifs risque d'être longue puisqu'elle peut faire intervenir des racines qui n'engendreront pas de consensus. La méthode des résidus semble mieux adaptée à condition de recenser les variables biformes et de répartir les racines de la représentation allégée selon deux classes d'équivalence modulo "... a des variables génératrices biformes".

Au fur et à mesure que la recherche progresse la première classe est subdivisée en deux sous-classes, eu égard à la présence des variables génératrices biformes non traitées.

8.311.2 Résidus. Consensus

Pour une même variable génératrice biforme, les résidus relatifs à une forme donnée de cette variable ne peuvent être que différents. Dans le cas contraire les racines seraient ou égales, ou liées par la relation d'ordre ce qui est absurde.

Le calcul des consensus est équivalent à la composition primaire de deux radicaux. Lorsque deux résidus relatifs aux deux formes d'une même variable sont égaux : les racines génératrices sont adjacentes (cf. <C-A710>).

8.311.3 Racine première

Toute racine génératrice de la forme allégée est racine première pour autant qu'elle ne soit pas antérieure à un consensus, engendré tant par les racines génératrices de la forme allégée que par des consensus racines premières.

Un consensus, engendré par des racines génératrices de la forme allégée, est racine première de celle-ci si il n'est antérieur à aucun autre consensus ou racine de cette même forme.

8.311.4 Essentialité

Conjointement à la recherche des racines premières, les racines de l'expression sont classées selon leur degré d'essentialité : essentielles ou non-essentielles proprement dites (cf. A8.23). Ce classement se fait par une recherche a contrario (ou par l'absurde).

Les racines génératrices sont considérées a priori essentielles. Nous dirons qu'elles sont quasi-essentielles. Une racine quasi-essentielle est non-essentielle proprement dite si elle égale un consensus engendré.

Un consensus engendré par des racines quasi-essentielles est a priori une racine non-essentielle proprement dite. Cette assertion reste a fortiori vraie si la racine quasi-essentielle est non-essentielle.

Tout consensus postérieur à une racine quasi-essentielle est racine quasi-essentielle. Il n'existe aucun radical, engendré par des racines quasi-essentielles ou non-essentielles (connues), postérieur à la racine donc a fortiori au consensus. Ce dernier prend donc la place de la racine quasi-essentielle dans la liste des racines quasi-essentielles de l'expression.

La figure 8.7 montre un exemple d'évolution d'essentialité de racines premières.

8.312 Description générale

Après avoir recherché (figure 8.3) la liste des variables biformes de la forme allégée, les résidus par rapport à chacune des formes de la variable biforme de plus faible rang sont calculés et enregistrés dans la moitié gauche du tableau EQUATI. Les racines génératrices sont réparties en trois classes eu égard aux types de variables :

- biformes traitées,
- biformes non traitées,
- monoformes,

et constituent l'ensemble des racines quasi-essentielles.

Les deux représentations, constituées par les résidus relatifs à chacune des formes de la variable, sont composées selon la loi primaire. Le résultat en est la liste des consensus relatifs à la variable biforme. Elle est placée dans la moitié droite du tableau EQUATI. Cette composition s'effectue de telle sorte qu'aucun consensus de la liste ne soit égal à un autre consensus ou ne lui soit antérieur.

Ils sont alors comparés aux racines quasi-essentielles :

- en cas d'égalité : la racine quasi-essentielle est supprimée de la liste;
- en cas de relation d'ordre (consensus, racine) : le premier est supprimé;
- en cas de relation d'ordre (racine, consensus) : la première est supprimée et le consensus est promu racine quasi-essentielle;

Dans les deux premiers cas et dans le cas général, le consensus comparé est rangé dans la liste des racines non-essentielles proprement dites, eu égard aux variables biformes qu'il est susceptible d'avoir pour variables génératrices.

Le traitement va reprendre avec la variable suivante mais avec une variante : les résidus des racines non-essentielles interviennent pour le calcul des consensus.

Il cesse après que la dernière variable biforme ait été considérée.

8.313 Représentation complète

La représentation complète est enregistrée dans le tableau EXPRES sous la forme de deux listes :

- L_E constituée par toutes les racines essentielles de la représentation;
- L_N ensemble des racines non-essentielles proprement dites.

Chacune de ces listes est divisée en deux sous listes (L_{E1}, L_{E2}, L_{N1}, L_{N2}), eu égard à l'existence ou non des variables biformes dans les racines.

8.32 Recherche d'une représentation première

8.321 Principes

Les éléments quasi-essentiels ont été reconnus lors de la recherche de la représentation complète. La recherche d'une forme première se ramène à :

- la promotion d'inessentialité des éléments quasi-essentiels consensus d'éléments quasi-essentiels;

- la promotion des racines non-essentielles proprement dites.

8.321.1 Racines inessentielles, pseudo-inessentielles

Les racines inessentielles seront les racines non-essentielles proprement dites, consensus des racines essentielles.

Les racines pseudo-inessentielles seront les racines non-essentielles consensus de racines essentielles, de racines pseudo-essentielles, de racines inessentielles ou pseudo-inessentielles.

8.321.2 Racine pseudo-essentielle. Critère de promotion

Les racines non-essentielles étant reconnues, nous rendrons progressivement pseudo-essentielles les racines qui

- ont le minimum de variables génératrices;
- engendrent le plus grand nombre de consensus avec les racines essentielles, pseudo-essentielles ou inessentielles et qui sont racines non-essentielles;
- engendrent le plus grand nombre de consensus avec les autres racines premières.

8.322 Description générale

Les variables biformes ne constituent le même ensemble que celui établi à partir de la représentation allégée que dans le cas le plus défavorable. Dans le cas général, elles en constituent un sous-ensemble. Les racines sont classées selon l'existence de ces variables. Nous pouvons cette fois calculer les consensus par la méthode d'itération.

La recherche d'une forme première va consister en (figure 8.4) :

- recherche des racines inessentielles;

- catégorisation des racines non-essentielles : nombre de variables, nombre de consensus engendrés;
- promotion de la racine non-essentielle adéquate, ou la première de la catégorie, en racine pseudo-essentielle;
- recherche des racines pseudo-inessentielles;
- réarrangement de la liste des racines non-essentielles;
- promotion d'une nouvelle racine non-essentielle ... etc.

Elle cesse lorsqu'il n'existe plus de racine non-essentielle.

8.4 DESCRIPTION FONCTIONNELLE

Le sous-programme TRADUC, qui assurait le contrôle de l'analyse de reconnaissance, assure aussi celui de l'analyse de réduction. Cette dernière est faite (figure 8.5) à l'aide des sous-programmes :

- EXECUT (EXECUTion) qui régit toute la recherche de la représentation allégée;
- NOCTUR (NOCTURne) qui permet l'obtention des représentations complète et première.

8.41 Recherche de la forme allégée

Le sous-programme EXECUT effectue (v. figure 8.6) la recherche amont et la recherche aval. Il procède à la complémentation des primaires, à la détection du type d'opération à effectuer; il gère donc les valeurs de NEGATI. Lorsqu'une composition primaire ou secondaire doit être effectuée, il appelle le sous-programme CHORAL.

Le programme CHORAL a pour charge :

- la détermination du type des opérandes : racine nulle, racine unité, radical;
- la commande du traitement des compositions (v. ci-après);
- le marquage du type de résultat.

8.411 Comparaison de racines

La recherche de la représentation allégée et, de ce fait, les compositions de représentations allégées implique la recherche de

- l'égalité entre les deux racines α et β ;
- la relation d'ordre $\alpha \dashv \beta$ ou $\beta \dashv \alpha$;
- la relation d'adjacence;
- la relation d'ordre entre le consensus engendré par α et β et les racines α et β . Le sous-programme CANON1 est conçu pour ces recherches, avec possibilité de n'effectuer que les deux premières exclusivement.

8.412 Composition secondaire

Dans le cas de la composition secondaire de deux racines, le sous-programme PASDE2 (PAS DE DEUX) l'effectue avec l'aide de CANON1.

Dans les autres cas il procède, assisté de CANON1, à la recherche de la représentation allégée. Cette recherche consiste en fait en la création de la représentation allégée d'un radical avec une racine de l'autre radical. Nous avons décrit en Annexe 8 l'algorithme de cette recherche. Les principes "machine" ont été donnés au chapitre précédent.

L'accès principal du sous-programme est réservé à la création d'une forme allégée, alors que l'entrée secondaire POLKA est spécialisée pour la composition secondaire de deux radicaux.

8.413 Composition primaire

8.413.1 De deux racines

Le sous-programme DUO procède à cette composition des deux racines, eu égard au codage machine choisi.

8.413.2 D'au moins un radical

Le sous-programme TRANS1 (TRANSposition 1) assure la composition primaire d'un radical et d'une racine ou de deux radicaux (entrée secondaire TRANS2).

Ces deux compositions consistent en :

- la recherche des racines compositions partielles primaires, avec l'aide du sous-programme DUO;
- la création de la forme allégée correspondante (entrée PASDE2);
- la composition secondaire avec la forme allégée résultat partiel précédent, par appel de POLKA.

8.42 Recherche des représentations complète et première

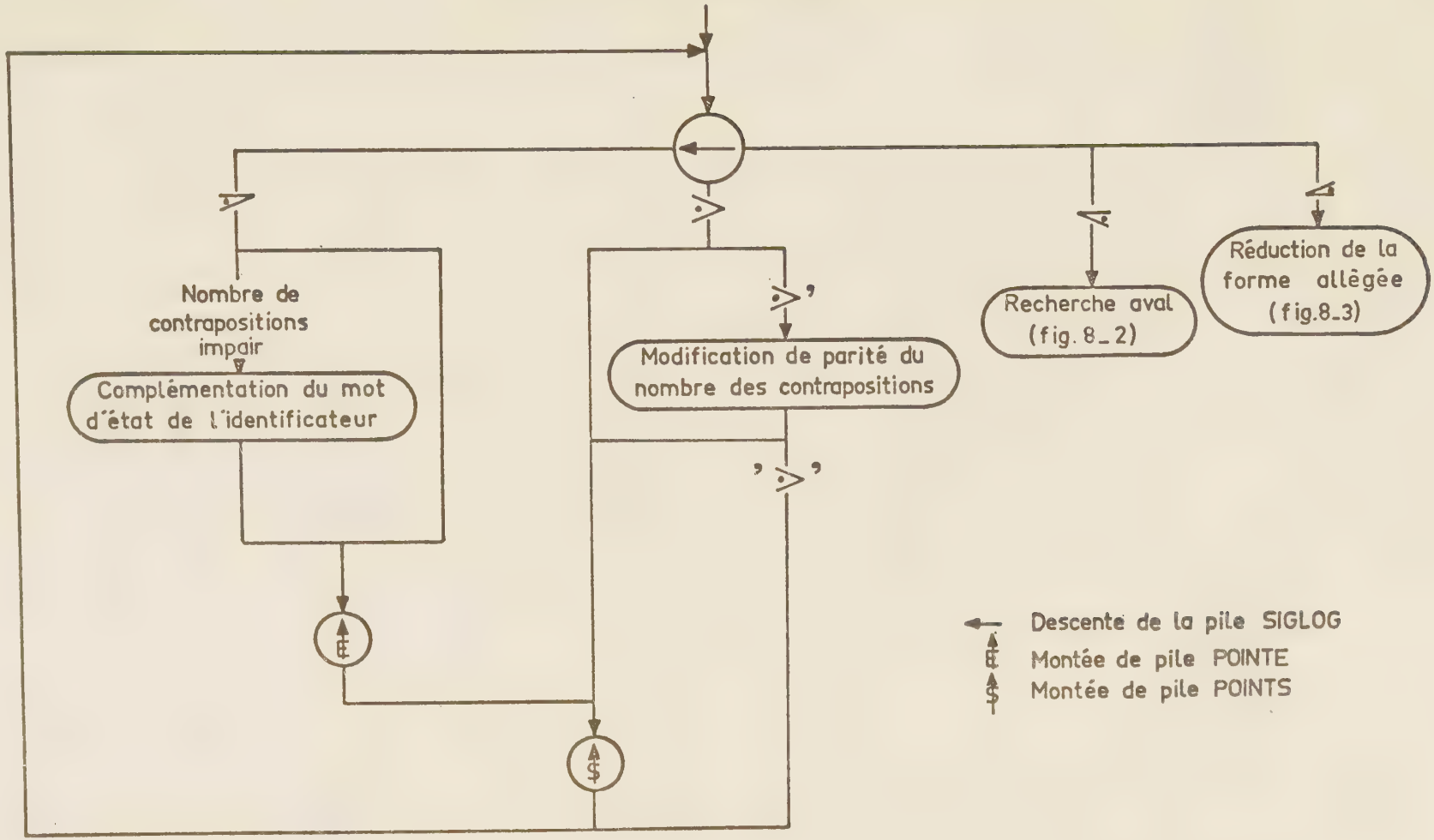
La figure 8.5 montre que le sous-programme NOCTUR (NOCTURne) est chargé de cette recherche, assisté du sous-programme BARCAR (BARCARolle) pour ce qui est de la forme première.

Les sous-programmes, DUO et CANON1, sont les seuls autres sous-programmes qui interviennent. Le premier pour la recherche des variables biformes et le calcul des consensus par la méthode des résidus. Le second est appelé pour la comparaison des résidus lors du calcul des consensus par la méthode des résidus (NOCTUR), pour le calcul des consensus par la méthode d'itérations successives (BARCAR) ou pour la comparaison - égalité, ordre - entre les racines (BARCAR et NOCTUR). Les figures 8.3 et 8.4 permettent, par référence aux fonctions, de voir à quels instants ces appels se produisent.

PROGRAMME CHOPIN

CHAPITRE 8

ANALYSE DE REDUCTION



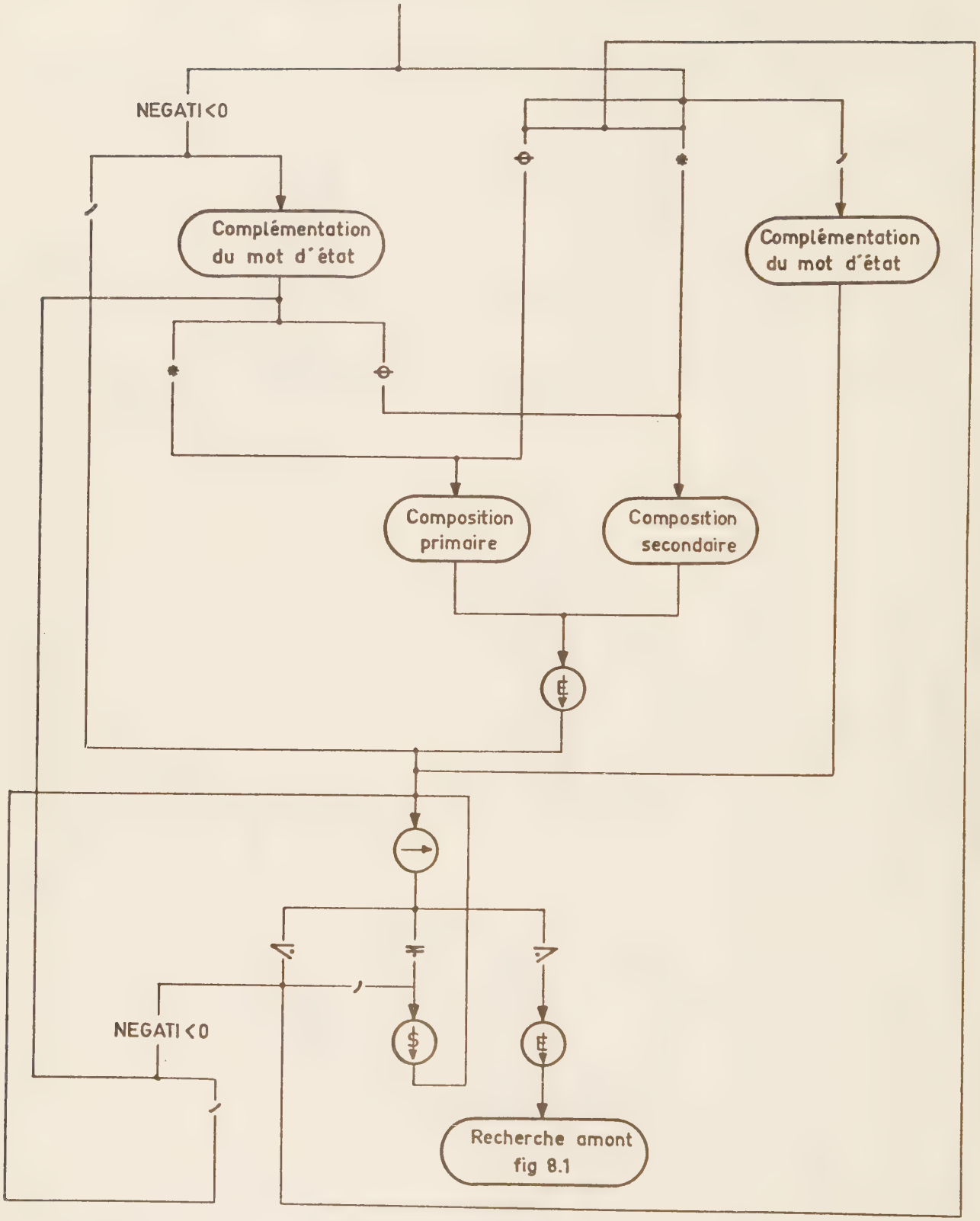
SCHEMA DE LA RECHERCHE AMONT

FIG. 8_1

PROGRAMME CHOPIN

CHAPITRE 8

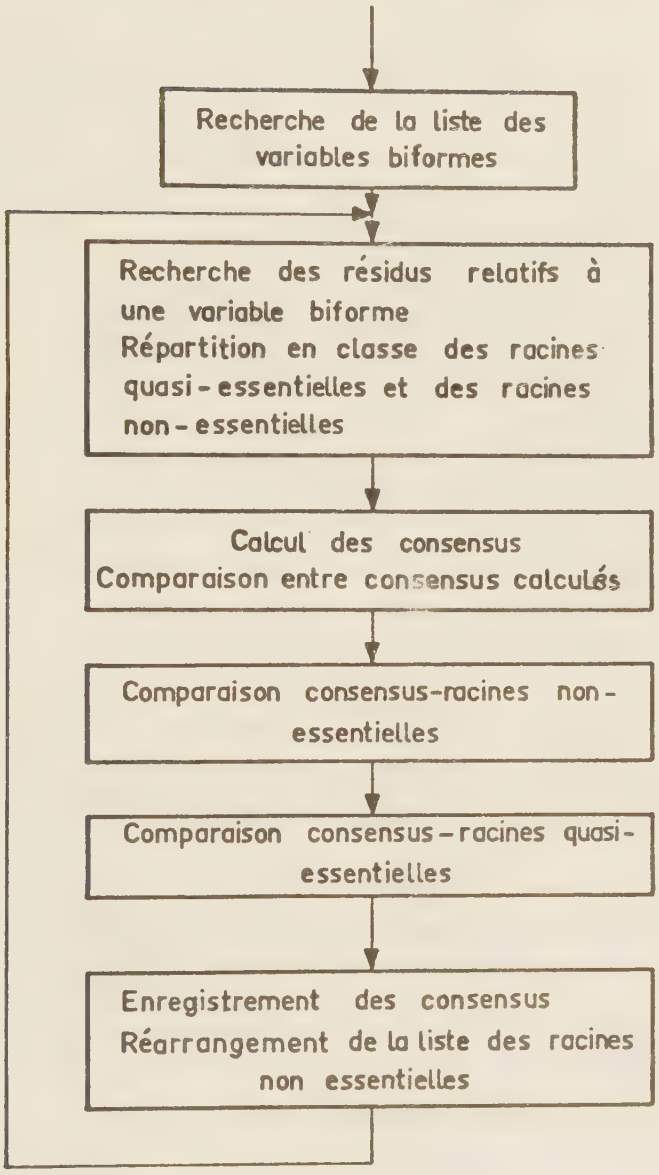
ANALYSE DE REDUCTION



RECHERCHE AVAL

FIG. 8_2

PROGRAMME CHOPIN
CHAPITRE 8
ANALYSE DE REDUCTION



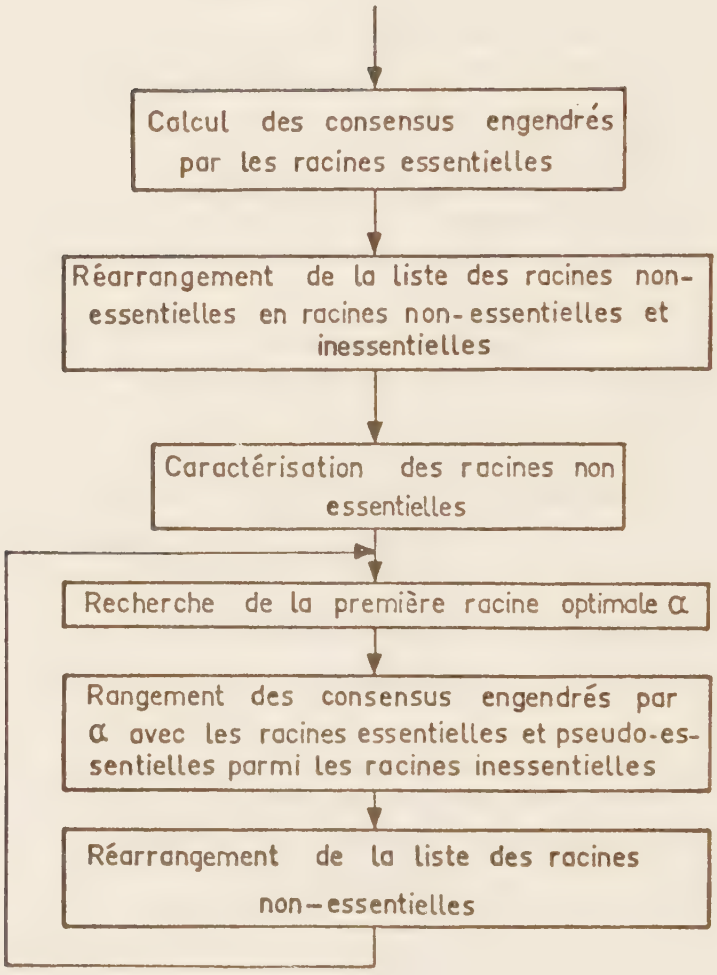
RECHERCHE DE LA REPRESENTATION COMPLETE

FIG_8_3

PROGRAMME CHOPIN

CHAPITRE 8

ANALYSE DE REDUCTION



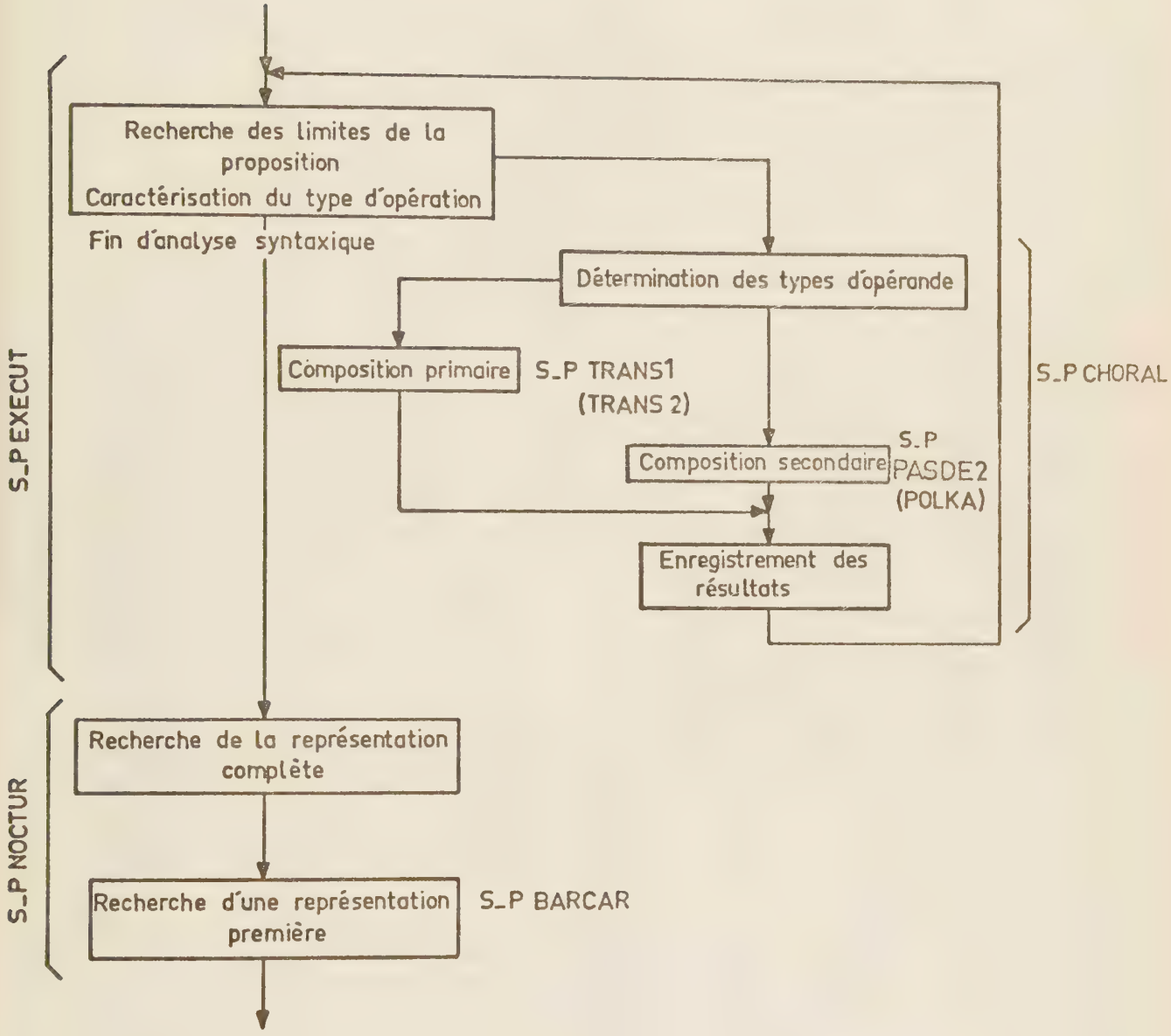
RECHERCHE D'UNE REPRESENTATION PREMIERE

FIG. 8_4

PROGRAMME CHOPIN

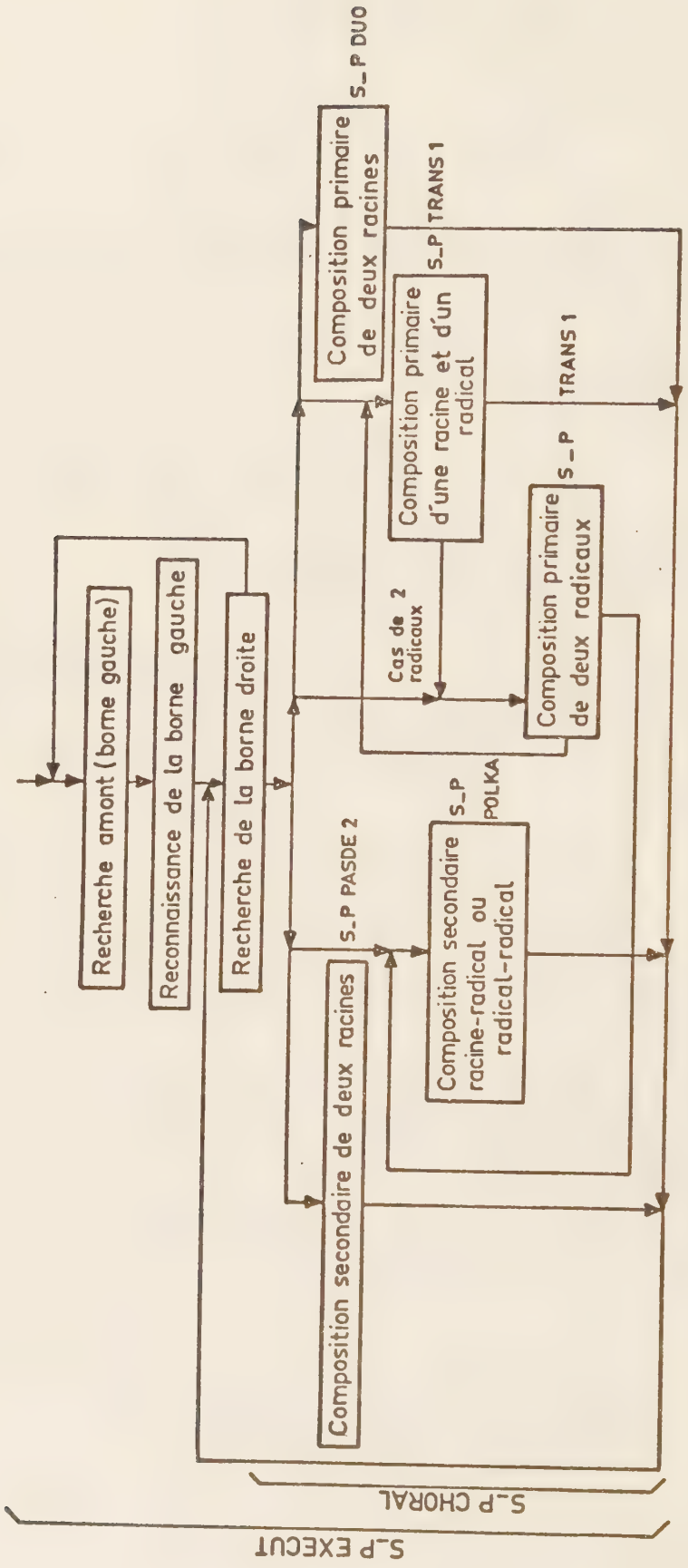
CHAPITRE 8

ANALYSE DE REDUCTION



ORGANISATION FONCTIONNELLE

FIG. 8_5



ORGANISATION FONCTIONNELLE DE LA
RECHERCHE D'UNE REPRESENTATION ALLEGEE

FIG. 8_6

PROGRAMME CHOPIN
CHAPITRE 8
ANALYSE DE REDUCTION

	N°	racines premières	consensus, d'ordre 2, des racines numéro	type de racine première	type après promotion	Ordre des promotions	représentation première
a'•b'•c'•d	1	a'•b'•c'•d		essentielle			x
a'•b•c•e'	2	b•c•d•e	8,10 - 9,10 -	Non essentielles proprement dites	pseudo-essentielle	2	x
a•b•c•d'	3	a•c'•d•e	6,8 -		pseudo-essentielle	1	x
b•c•d•e	4	a•b•c•d'	7,9 -		pseudo-essentielle	4	x
a•c'•d•e	5	a'•b•c•e'	7,10 -		pseudo-essentielle	3	x
v.figure 3.4	6	b'•c'•d•e	1,3 -		pseudo-inessentielle	1	
	7	b•c•d'•e'	4,5 -		pseudo-inessentielle	4	
	8	a•b•d•e	2,3 - 3,9 -		pseudo-inessentielle	2	
	9	a•b•c•e	2,4 - 4,8		pseudo-inessentielle	4	
	10	a'•b•c•d	2,5 -		pseudo-inessentielle	3	

EXEMPLE D'EVOLUTION DE L'ESSENTIALITE DES RACINES PREMIÈRES

FIG. 8_7

C O N C L U S I O N

Tel que nous le définissons au cours de cette étude, l'allégement des représentations, combiné à la méthode des consensus nous semble être la meilleure méthode de traitement d'expressions booléennes du type de celles rencontrées couramment en pratique.

La formalisation du langage nous permet d'envisager le traitement de ces expressions en première ou seconde forme, indifféremment.

D'autre part, la présentation des données, utilisée tant en entrée qu'en sortie, facilite son emploi.

Des améliorations, sous forme "d'interfaces programmées", sont envisageables :

- introduction des données en représentation numérique. Cette interface devrait intervenir, après la caractérisation d'une grammaire de reconnaissance, au niveau de l'analyse de réduction.

- introduction des données sous forme de description technologique. Une telle "interface" existe. Elle permet, à partir d'une description d'opérateurs tels que NOR, NAND, OR, AND etc...., d'obtenir une expression littérale en représentation brute. Nous avons donné un exemple d'utilisation de cette interface suivie du traitement de l'expression obtenue. Une modification consisterait en une description à partir des types technologiques. Signalons que l'interface actuelle est compatible avec le langage de description du programme, CHAMBOR, de simulation du fonctionnement d'ensembles numériques et logiques. Ce programme est un sous-ensemble du système KOALA.

- introduction d'opérations ou lois de composition complexes. Nous pensons aux opérations de la logique combinatoire telles que OU exclusif, coïncidence logique, majorité etc....

Une amélioration, subordonnée à une recherche théorique, consisterait en l'introduction d'opérations, ou de fonctions de logique séquentielle : mémoires, bascules bistables, registres, compteurs etc..... Nous pensons pouvoir, dans quelque temps, y apporter une solution partielle.

Citons avant de terminer quelques applications possibles du programme CHOPIN.

Une première application se situe dans le domaine pédagogique. CHOPIN permet d'obtenir, rapidement, la forme complète et une forme première d'une expression dont le nombre de variables génératrices peut atteindre trente deux. Il peut être utilisé tant pour engendrer de nouveaux exercices que pour permettre à l'étudiant de vérifier la validité de solutions qu'il envisage pour un problème posé. Il permet de vérifier l'équivalence de diverses expressions. Toutes les applications, que nous citons par la suite, ont leur répercussion au stade de l'enseignement.

Une seconde application, d'ordre pratique, est d'obtenir la (ou les) expression d'un circuit technologique de forme complexe. En plus de la caractérisation de définition de ce circuit, il nous est possible de caractériser le circuit pour des applications ultérieures:

- comparaison avec d'autres circuits;
- symbolisation du circuit pour la simulation du fonctionnement d'ensembles complexes;
- symbolisation du circuit pour la recherche de séquence de test d'ensembles où figure ce circuit;
- ... etc.

Une troisième application, au niveau de la synthèse, permet de vérifier que les représentations technologiques correspondent bien aux définitions. Cette vérification est utile pour la synthèse du type "multi-fonction", où les racines communes à plusieurs expressions sont recherchées. Elle est aussi un aboutissement de la synthèse de circuits séquentiels. Celle-ci fournit en résultat des expressions d'éléments à retard (type D) ou d'éléments mémoires (RS). Une étude de ces expressions permet de distinguer des types d'éléments technologiques plus élaborés : JK, T, J_TK_T, D_G, D_T, $J_GK_G$etc.... Elle ne peut être envisagée qu'à partir des représentations complètes des "fonctions" calculées.

Une quatrième application ressort de l'analyse séquentielle de circuits existants. A partir des représentations obtenues, il sera possible de reconstituer les matrices d'excitation, puis les matrices de phases d'un ensemble séquentiel. L'encombrement mémoire, nécessaire pour un tel traitement, limite cette application aux circuits dont le nombre des variables, tant en types d'entrées que variables secondaires et de sortie, ne dépasse pas quinze.

Remarquons, enfin, l'impact psychologique que représente un tel programme, l'importance qu'il revêt, en ce qui nous concerne, quant à l'avenir du système KOALA. Une des difficultés majeures que nous rencontrons dans l'effort de

diffusion, de vulgarisation des moyens qu'il offre réside dans le langage d'entrée. Un langage qui demande un trop gros effort de réflexion, une transposition proche du langage machine est rebutant. L'utilisateur potentiel qui ne doit pas être forcément spécialisé aime à retrouver et à utiliser un mode de description qui lui est familier. Nous pensons qu'en lui offrant la possibilité de représenter des expressions booléennes sous la forme classique réalise une partie de cette demande. Tout comme pour le programme CHAMBOR, elle réduira l'opposition inévitable de l'électronicien à l'utilisation de moyens qui semblent destinés à réduire ses privilèges mais, qui en fait le déchargent de la partie, non noble, de sa tâche.

B I B L I O G R A P H I E

- GRAMMAIRES - LANGAGES -

N. CHOMSKY
On certain formal properties of grammars
Inf. Cont. 2 (1959).

N. CHOMSKY
Formal properties of grammars
Handbook of Mathematical Psychology, Wiley and Sons, New-York (1963).

N. CHOMSKY, G.A. MILLER
Analyse formelle des langages naturels
Gauthier-Villars, Paris (1968).

P. DEBRAINE
Machines de traitement de l'information (tome 3 en préparation)
Masson et Cie, Paris.

R.W. FLOYD
Syntactic analysis and operator precedence
J.A.C.M., volume 10, p. 316, (juillet 1963).

J.M. FOSTER
Automatic syntactic analysis
Macdonald, Londres (1970).

M. GROSS, A. LENTIN
Notions sur les grammaires formelles
Gauthier-Villars, Paris (1967).

J.C. HELIARD
Génération automatique de processeurs de syntaxe à partir d'une N.B.F.
(thèse docteur 3ème cycle)
Faculté des Sciences, Paris-Sud, Orsay (janvier 1972).

H. KATZAN, Jr.
Advanced Programming
Van Nostrand Reinhold Company, New-York.

D.F. MARTIN
Boolean matrix methods for the detection of simple precedence grammars
C.A.C.M., volume 11, p. 685 (octobre 1968).

R. TARBOURIECH
Compilateur ALGOL - Analyse syntaxique et sémantique
(thèse docteur 3ème cycle)
Faculté des Sciences, Paris (mars 1972).

S. WARSHALL
A theorem an boolean matrix
J.A.C.M., volume 9, p. 11 (janvier 1962).

N. WIRTH, H. WEBER
EULER - A generalisation of ALGOL and its formal definition
C.A.C.M., volume 19, p.13 (janvier 1966) et p. 89 (février 1966).

- L O G I Q U E -

A. BLANCHARD
Eléments de commutation générale - Applications aux systèmes de téléphonie automatique
Eyrolle, Paris (1963).

M. CARVALLO
Principes et applications de l'analyse booléenne
Gauthier-Villars, Paris 1965).

M. CARVALLO
Monographie des treillis et algèbre de Boole
Gauthier-Villars, Paris (1966).

I.C. CAVADIA
An algorithm for the formal calculus of boolean expressions
R.I.R.O. - B3, p. 65 (septembre 1971).

P. DEBRAINE
Machines de traitement de l'information (tome 1)
Masson et Cie, Paris (1967).

E.J. McCLUSKEY.Jr.
Minimization of boolean functions
Bell Syst. Techn. Journal, 35, p. 1417 (novembre 1956)

P. NASLIN
Circuits logiques et automatismes à séquences
Dunod, Paris (1965).

P. TISON
Théorie des consensus - Algorithme de recherche des bases premières
(Algèbre de Boole et machines logiques, J. KUNTZMANN, P. NASLIN), Dunod, Paris (1967).

R.L. VALLEE
Analyse binaire (tome 1)
Masson et Cie, Paris (1970).

A N N E X E S

A N N E X E O

- EXEMPLES -

Afin de présenter le plus grand nombre d'exemples possible, cette annexe et les figures qui l'accompagnent sont agencées comme suit :

- description générale de la présentation des résultats;
- illustration de cas particuliers (vocabulaire terminal, complémentation, résultat trivial);
- exemples que nous appelons "classiques", empruntés à la bibliographie (notamment à l'ouvrage de Monsieur DEBRAINE);
- exemples qui correspondent à un circuit existant et qui fait intervenir un nombre de variables relativement élevé pour être difficilement traités manuellement.

AO.1 PRESENTATION DES RESULTATS

Pour un sous-module de données, c'est-à-dire un ensemble d'expressions à traiter avec le même vocabulaire terminal, les résultats sont présentés comme suit :

- première page : liste des opérateurs et des symboles choisis par l'utilisateur; liste des variables réparties selon leur type. Les figures AO.1 et AO.2 en sont deux exemples.
- pages suivantes : expression fournie en entrée (transposition textuelle des cartes correspondantes); résultat de la réduction, à savoir :

  - messages d'erreur,

  ou

  - expression en représentation première, suivie de la sous-liste qui rend cette représentation complète.

Nous n'avons reproduit que ces pages pour les exemples traités. La liste des opérateurs choisie est celle donnée par la figure AO.2.

AO.2 CAS PARTICULIERS

AO.21 Vocabulaire terminal

La figure AO.3 reproduit les cartes de description du vocabulaire terminal qui a permis d'obtenir les résultats reproduits figure AO.1.

A0.22 Complémentation

La figure A0.4 montre quelques exemples simples de complémentation. Ces exemples ne font apparaître que les expressions d'entrée et les expressions résultats. Ils avaient pour but de vérifier la complémentation.

A0.23 Résultats triviaux

La figure A0.5 reproduit, dans les mêmes conditions que précédemment, des exemples destinés à faire apparaître les résultats triviaux.

A0.3 EXEMPLES CLASSIQUES

Les figures suivantes (A0.6 à A0.11) reproduisent les résultats obtenus en traitant des exemples empruntés à la bibliographie. La figure A0.12 reproduit les diagrammes de KARNAUGH de quelques uns de ces exemples. Le lecteur pourra ainsi vérifier et comparer les résultats.

A0.4 CIRCUIT REEL

Les résultats (figures A0.13 à A0.17) qui suivent sont relatifs au circuit dont le schéma est donné figure 1.1.

A0.41 Expression de Fo

L'expression Fo a été traitée en première et seconde forme. Dans ce dernier cas l'expression d'entrée doit être plus "fortement parenthésée", nous avons pu alors apprécier l'avantage que présente l'insertion des blancs. La figure A0.17 représente le diagramme de KARNAUGH de Fo, en première forme. Il est facile de confronter les résultats.

A0.42 Expression de F_1

Cette expression n'a été traitée qu'en première forme. Ici encore, le diagramme de KARNAUGH reproduit figure A0.1 nous a permis de vérifier la validité du résultat obtenu.

A0.43 Expression de F_2

Cette expression traitée en première forme a été vérifiée en plaçant le circuit sur une maquette d'essai. Pour chaque racine, nous avons couvert la représentation canonique et vérifié ainsi la validité du résultat.

A0.5 TEMPS D'EXECUTION

Le tableau ci-après résume les caractéristiques d'exécution dans les cas précédents. La procédure utilisée est FORTHG.

Nous devons y ajouter les valeurs suivantes :

- temps de compilation total : 41,02 s.
- temps de chaînage : 1 s.
- encombrement mémoire lors de l'exécution : 70 k octets.
- nombre de cartes FORTRAN : environ 4.000 cartes.
- nombre d'instructions FORTRAN : 2.668.
- nombre de cartes binaires : 1.181 cartes.

Expression	Nombre de variables génératrices	Nombre de racines de la représentation : première	Nombre de racines de la représentation : complète	Forme	Temps d'exécution en secondes
Fo	8	12	27	1	0,86
	8	12	27	2	
F_1	10	40	75	1	2,51
F_2	12	95	175	1	16,91
	16	8	12	1	5,21
		72	100		
Autres exemples		temps total d'exécution			1,20

**********LISTE DES OPERATEURS**********

EGAL =
PAS *
ET .
OU +

**********LISTE DES VARIABLES**********

ENTREE	SECONDAIRE	SECONDAIRE-SORTIE	SORTIE
AA	SEC0	SCR0	SOR0
BB	SEC1	SCR2	SOR1
CC	SEC2	SCR1	
DD			
EFGH			
IJ			

1ère PAGE RESULTAT DES TRAITEMENTS DES "MODULES DONNEES"

FIG. A0-1

1ère PAGE RESULTAT DES TRAITEMENTS DES "MODULES DONNEES"

FIG. A0.2

```
FIN DE LISTE D'ENTREE
VSOR  SORO SOR1 SORO SOR1 SOR2
VSEC  SECO SEC1 SEC2 SORO SOR1 SOR2
VENT  AA BB CC DD EFGH IJ
OPER EGAL = ET . OU + PAS *
```

FIG.A0.3
CARTES DU SOUS_MODULE VOCABULAIRE

PROGRAMME CHOPIN

ANNEXE 0

EXEMPLES

```
********** EQUATION DONNEE EN ENTREE **********

 EQ00 Q=(A.B)'

********** EXPRESSION REDUITE DE L'EQUATION  EQ00***
LE RESULTAT EST DONNE EN PREMIERE FORME.

    EQ00 Q=A'+B'

********** LA REPRESENTATION EST COMPLETE **********

********** EQUATION DONNEE EN ENTREE **********

 EQ01 Q=(A.B)'                                    INTR

********** EXPRESSION REDUITE DE L'EQUATION  EQ01***
LE RESULTAT EST DONNE EN SECONDE FORME.

    EQ01 Q=(A'+B')

********** LA REPRESENTATION EST COMPLETE **********

********** EQUATION DONNEE EN ENTREE **********

 EQ02 Q=(A.B)''

********** EXPRESSION REDUITE DE L'EQUATION  EQ02***
LE RESULTAT EST DONNE EN PREMIERE FORME.

    EQ02 Q=A.B

********** LA REPRESENTATION EST COMPLETE **********

********** EQUATION DONNEE EN ENTREE **********

 EQ03 Q=(A.B)''                                   INTR

********** EXPRESSION REDUITE DE L'EQUATION  EQ03***
LE RESULTAT EST DONNE EN SECONDE FORME.

    EQ03 Q=A.B

********** LA REPRESENTATION EST COMPLETE **********
```

FIG. A0.4a.

COMPLEMENTATION

(Exemples simples)

```
                    ********** EQUATION DONNEE EN ENTREE **********
 EQ04 Q=(A+B)'
                  ********** EXPRESSION REDUITE DE L'EQUATION  EQ04***
LE RESULTAT EST DONNE EN PREMIERE FORME.
     EQ04 Q=A'.B'
********************* LA REPRESENTATION EST COMPLETE *********************

                    ********** EQUATION DONNEE EN ENTREE **********
 EQ05 Q=(A+B)'                                                    INTR
                  ********** EXPRESSION REDUITE DE L'EQUATION  EQ05***
LE RESULTAT EST DONNE EN SECONDE FORME.
     EQ05 Q=A'.B'
********************* LA REPRESENTATION EST COMPLETE *********************
```

FIG.A0.4b
COMPLEMENTATION
(Exemples simples)

ANNEXE 0

EXEMPLES

```
********** EQUATION DONNEE EN ENTREE ********

 UNIT Q=A+A'

********** EXPRESSION REDUITE DE L'EQUATION  UNIT**

LE RESULTAT EST DONNE EN PREMIERE FORME.

    UNIT Q=UN

********** EQUATION DONNEE EN ENTREE ********

 UNIT Q=A+A'                                          INTR

********** EXPRESSION REDUITE DE L'EQUATION  UNIT**

LE RESULTAT EST DONNE EN SECONDE FORME.

    UNIT Q=UN

********** EQUATION DONNEE EN ENTREE ********

 ZERO Q=A'.A

********** EXPRESSION REDUITE DE L'EQUATION  ZERO**

LE RESULTAT EST DONNE EN PREMIERE FORME.

    ZERO Q=ZERO

********** EQUATION DONNEE EN ENTREE ********

 ZERO Q=A'.A                                          INTR

********** EXPRESSION REDUITE DE L'EQUATION  ZERO**

LE RESULTAT EST DONNE EN SECONDE FORME.

    ZERO Q=ZERO
```

FIG. A0.5

EXPRESSIONS TRIVIALES

```
********** EQUATION DONNEE EN ENTREE ********

 DEBO X= A.B'.C.D' + A.B'.C.D + A.B.C.D'

                ********** EXPRESSION REDUITE DE L'EQUATION  DEBO**

LE RESULTAT EST DONNE EN PREMIERE FORME.

    DEBO X=A.B'.C+A.C.D'

********** LA REPRESENTATION EST COMPLETE **********
                ********** EQUATION DONNEE EN ENTREE ********

 DEBO X=(A.B'.C.D')+(A.B'.C.D)+(A.B.C.D')                    INTR

                ********** EXPRESSION REDUITE DE L'EQUATION  DEBO**

LE RESULTAT EST DONNE EN SECONDE FORME.

    DEBO X=A.(B'+D').C

********** LA REPRESENTATION EST COMPLETE **********
                ********** EQUATION DONNEE EN ENTREE ********

 DEBI X< A'.B'.C' + A'.B'.C + A'.B.C' + A'.B.C + A.B'.C

                ********** EXPRESSION REDUITE DE L'EQUATION  DEBI**

LE RESULTAT EST DONNE EN PREMIERE FORME.

    DEBI X<A'+B'.C

********** LA REPRESENTATION EST COMPLETE **********
```

.FIG.AO.6
"EXEMPLES CLASSIQUES"
RESULTATS DU TRAITEMENT

PROGRAMME CHOPIN
ANNEXE 0
EXEMPLES

```
********** EQUATION DONNEE EN ENTREE **********

 DEB1 X<(A'.B'.C')+(A'.B'.C)+(A'.B.C')+(A'.B.C)+(A.B'.C)                    INTR

                ********** EXPRESSION REDUITE DE L'EQUATION  DEB1**********

LE RESULTAT EST DONNE EN SECONDE FORME.

    DEB1 X<(A'+C).(A'+B')

********** LA REPRESENTATION EST COMPLETE **********
                ********** EQUATION DONNEE EN ENTREE **********

 DEB1 X> A'.B'.C' + A'.B'.C + A'.B.C' + A'.B.C + A.B'.C +A.B.C'+A.B.C

                ********** EXPRESSION REDUITE DE L'EQUATION  DEB1**********

LE RESULTAT EST DONNE EN PREMIERE FORME.

    DEB1 X>A'+C+B

********** LA REPRESENTATION EST COMPLETE **********
                ********** EQUATION DONNEE EN ENTREE **********

 DEB1 X<(A'.B'.C')+(A'.B'.C)+(A'.B.C')+(A'.B.C)+(A.B'.C)+(A.B.C')+(A.B.C    INTR
     *)

                ********** EXPRESSION REDUITE DE L'EQUATION  DEB1**********

LE RESULTAT EST DONNE EN SECONDE FORME.

    DEB1 X<(A'+B+C)

********** LA REPRESENTATION EST COMPLETE **********
```

.FIG.AO.7
"EXEMPLES CLASSIQUES"
RESULTATS DU TRAITEMENT

PROGRAMME CHOPIN
ANNEXE 0
EXEMPLES

```
********** EQUATION DONNEE EN ENTREE **********
DEB2 X=(A+C+D).(A'+B+D).(A'+D').(A+B+D)


********** EXPRESSION REDUITE DE L'EQUATION  DEB2**********

LE RESULTAT EST DONNE EN PREMIERE FORME.

    DEB2 X=A'.D+A.B.D'+A'.B.C

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********

        B.C.D'

********** EQUATION DONNEE EN ENTREE **********
DEB2 X=(A+C+D).(A'+B+D).(A'+D').(A+B+D)                                  INTR


********** EXPRESSION REDUITE DE L'EQUATION  DEB2**********

LE RESULTAT EST DONNE EN SECONDE FORME.

    DEB2 X=(A+C+D).(A'+D').(B+D)

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********

        (A'+B)

********** EQUATION DONNEE EN ENTREE **********
DEB3 X=(A'+B'+C'+D').(A'+B'+C+D').(A'+B+C'+D').(A'+B+C'+D).(A'+B+C+D').
    *(A'+B+C+D).(A+B'+C+D).(A+B+C'+D).(A+B+C+D)


********** EXPRESSION REDUITE DE L'EQUATION  DEB3**********

LE RESULTAT EST DONNE EN PREMIERE FORME.

    DEB3 X=A'.D+A.B.D'+A'.B.C

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********

        B.C.D'
```

.FIG.AO.8
"EXEMPLES CLASSIQUES"
RESULTATS DU TRAITEMENT

PROGRAMME CHOPIN

ANNEXE 0

EXEMPLES

```
                    ********** EQUATION DONNEE EN ENTREE **********

 DEB3 X=(A'+B'+C'+D').(A'+B'+C+D').(A'+B+C'+D').(A'+B+C'+D).(A'+B+C+D').     INTR
     *(A'+B+C+D).(A+B'+C+D).(A+B+C'+D).(A+B+C+D)


                    ********** EXPRESSION REDUITE DE L'EQUATION  DEB3**********

LE RESULTAT EST DONNE EN SECONDE FORME.

     DEB3 X=(A'+D').(B+D).(A+C+D)

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********************************************************

          (A'+B)
                    ********** EQUATION DONNEE EN ENTREE **********
 DEB4 X=(A'.B'.C'.D)+(A'.B.C'.D)+(A'.B.C.D)+(A.B.C'.D')+(A.B.C'.D)+(A.B.
     *C.D')+(A.B.C.D)


                    ********** EXPRESSION REDUITE DE L'EQUATION  DEB4**********

LE RESULTAT EST DONNE EN PREMIERE FORME.

     DEB4 X=A'.C'.D+A.B+B.D

********** LA REPRESENTATION EST COMPLETE **********
                    ********** EQUATION DONNEE EN ENTREE **********
 DEB4 X=(A'.B'.C'.D)+(A'.B.C'.D)+(A'.B.C.D)+(A.B.C'.D')+(A.B.C'.D)+(A.B.     INTR
     *C.D')+(A.B.C.D)


                    ********** EXPRESSION REDUITE DE L'EQUATION  DEB4**********

LE RESULTAT EST DONNE EN SECONDE FORME.

     DEB4 X=(A'+B).(A+D).(B+C')

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********************************************************

          (B+D)
```

.FIG.AO.9

"EXEMPLES CLASSIQUES"

RESULTATS DU TRAITEMENT

ANNEXE 0

EXEMPLES

```
                    ********** EQUATION DONNEE EN ENTREE **********
DEB8 X=  A'.B'.C .( D'.E'.F + D'.E.F + C.E.F ) +  A'.B.C . ( D'.E'.F +    81
     * D.E'.F ) +  A'.B.C' . ( D'.E'.F' + D.E.F' + D.E'.F' ) +  A.B.C' .  82
     * ( D'.E.F' + D.E'.F' ) +  A.B'.C . ( D'.E'.F + D.E'.F )+D'.E'.F'.(  83
     *A'.B'.C'+A.B.C'+A.B.C)+A.B'.C.D'.E.F+(A'.B'.C'+A'.B.C').D'.E.F'+(A  84
     *.B.C'+A .B.C').D.E.F'+D.F.F.(A'.B.C+A.B'.C)+A'.B'.C.D.E'.F          85


                    ********** EXPRESSION REDUITE DE L'EQUATION  DEB8*******:

LE RESULTAT EST DONNE EN PREMIERE FORME.

    DEB8 X=B.C'.F'+A'.C.E'.F+A.B.D'.E'.F'+A'.C'.D'.F'+A'.C.D.F+B'.C.F

********** LA REPRESENTATION EST COMPLETE **********
                    ********** EQUATION DONNEE EN ENTREE **********
 CNS3 X<A.B.D'+A'.B.D+A'.B'.C+A'.C.D


                    ********** EXPRESSION REDUITE DE L'EQUATION  CNS3********

LE RESULTAT EST DONNE EN PREMIERE FORME.

    CNS3 X<A.B.D'+A'.B.D+A'.B'.C

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********

        A'.C.D
                    ********** EQUATION DONNEE EN ENTREE **********
 CNS3 X> A.C+A.D'+A'.B.C'+A'.B.D+B'.C+B'.D'+C.D+C'.D'+A.B'


                    ********** EXPRESSION REDUITE DE L'EQUATION  CNS3*******:

LE RESULTAT EST DONNE EN PREMIERE FORME.

    CNS3 X>C'.D'+C.D+A.B'+A.D'+A'.B.C+B'.C

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********

        B'.D'+A'.B.C'+A.C
```

.FIG.A0.10

"EXEMPLES CLASSIQUES"

RESULTATS DU TRAITEMENT

```
********** EQUATION DONNEE EN ENTREE **********

 CNS1 F=A.X+A.Y+B.X+B.Y+B.Z+X.Y'+X.Z'+X'.Y+Y.Z'

********** EXPRESSION REDUITE DE L'EQUATION  CNS1**********

LE RESULTAT EST DONNE EN PREMIERE FORME.

    CNS1 F=B.Z+X.Y'+X'.Y+Y.Z'+A.Y

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
**********************************************************

         A.X+B.Y+B.X+X.Z'



********** EQUATION DONNEE EN ENTREE **********

 CNS1 F=(A.X)+(A.Y)+(B.X)+(B.Y)+(B.Z)+(X.Y')+(X.Z')+(X'.Y)+(Y.Z')          INTR

********** EXPRESSION REDUITE DE L'EQUATION  CNS1**********

LE RESULTAT EST DONNE EN SECONDE FORME.

    CNS1 F=(X+Y+Z).(A+B+X'+Y'+Z').(B+X+Y)

********** LA REPRESENTATION EST COMPLETE **********
```

.FIG. AO .11
"EXEMPLES CLASSIQUES"
RESULTATS DU TRAITEMENT

PROGRAMME CHOPIN
ANNEXE 0
EXEMPLES

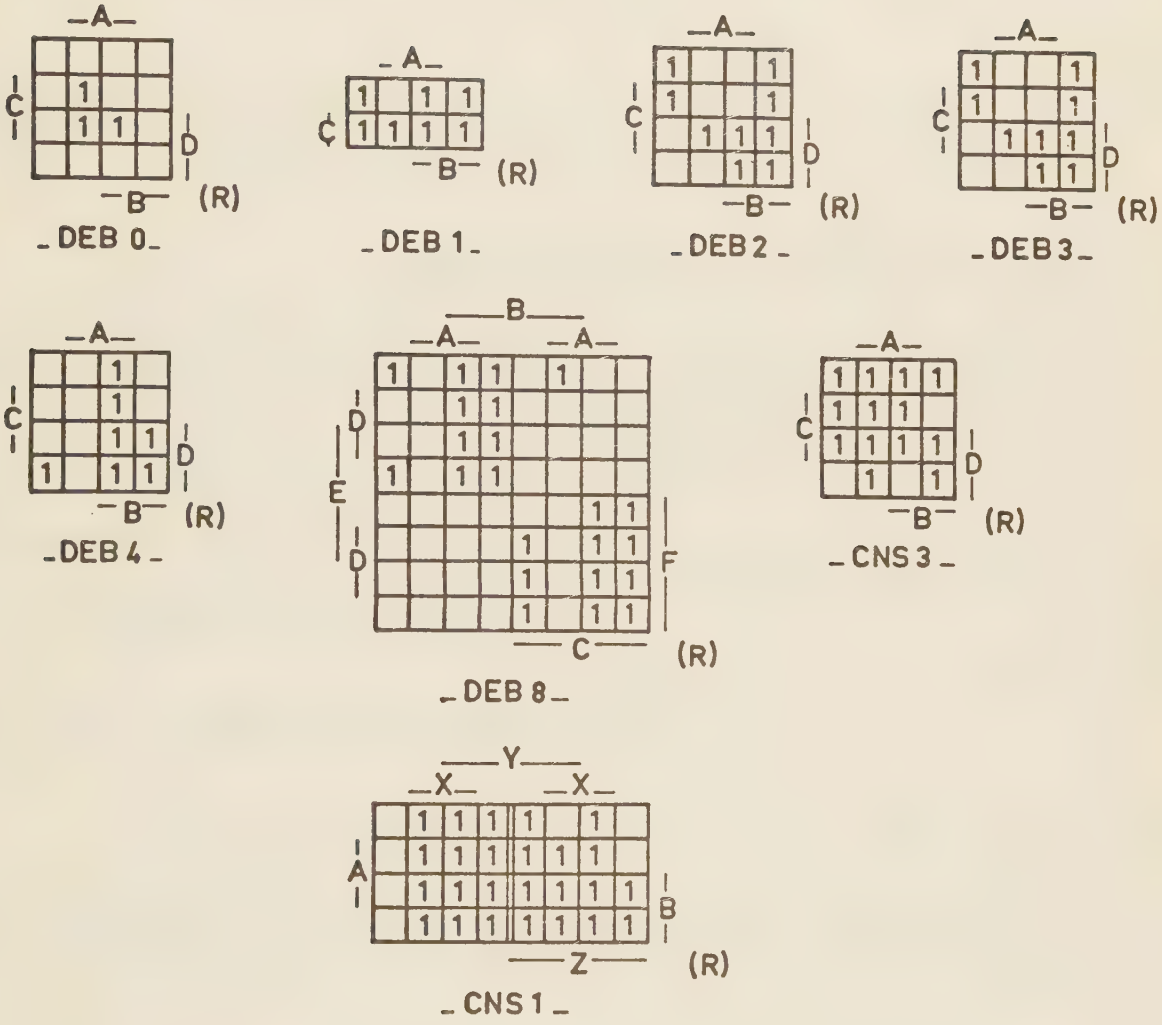


FIGURE A0_12
" EXEMPLES CLASSIQUES
DIAGRAMME DE KARNAUGH _

ANNEXE 0

EXEMPLES

********** EQUATION DONNEE EN ENTREE **********

```
 F0=(((B0.S3.A0+A0.S2.B ')'+(B0'.S1+B0.S0+A0)').((B0.S3.A0+A0.
*S2.B0')'.(B0'.S1+B0.S0+A0)')'+(M'.CN)').(((B0.S3.A0+A0.S2.B0'
*)'+(B0'.S1+B0.S0+A0)').((B0.S3.A0+A0.S2.B0')'.(B0'.S1+B0.S0+A0
*)')'.(M'.CN)')'
```

********** EXPRESSION REDUITE DE L'EQUATION **********

LE RESULTAT EST DONNE EN PREMIERE FORME.

```
F0=A0.B0.CN.M'.S3'+A0.B0.CN'.S3+A0.B0.M.S3+A0.B0'.CN.M'.S2'+A0.
B0'.CN'.S2+A0.B0'.M.S2+A0'.B0'.CN.M'.S1+A0'.B0'.M.S1'+A0'.B0'.C
N'.S1'+A0'.B0.CN'.S0'+A0'.B0.M.S0'+A0'.B0.CN.M'.S0
```

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :

```
**************************************************
```

```
CN'.S0'.S1'.S2.S3+M.S0'.S1'.S2.S3+CN.M'.S0.S1.S2'.S3'+A0.CN.M'.
S2'.S3'+A0.CN'.S2.S3+A0.M.S2.S3+B0'.CN.M'.S1.S2'+B0'.M.S1'.S2+B
0'.CN'.S1'.S2+B0.CN'.S0'.S3+A0'.CN'.S0'.S1'+B0.M.S0'.S3+A0'.M.S
0'.S1'+B0.CN.M'.S0.S3'+A0'.CN.M'.S0.S1
```

********** EQUATION DONNEE EN ENTREE **********

```
 F0=(                                                                  INTR
#   ((((B0.S3.A0)+(A0.S2.B0'))'+((B0'.S1)+(B0.S0)+A0)').(((B0.S3.A0
F001#)+(A0.S2.B0'))'.((B0'.S1)+(B0.S0)+A0)')')
F001#                                          +(M'.CN)').((((B0.S3.A0)+(
F002#A0.S2.B0'))'+((B0'.S1)+(B0.S0)+A0)').(((B0.S3.A0)+(A0.S2.B0'))'.((
F003#B0'.S1)+(B0.S0)+A0)')'.(M'.CN)')'
```

********** EXPRESSION REDUITE DE L'EQUATION **********

LE RESULTAT EST DONNE EN SECONDE FORME.

```
F0=(A0'+B0'+CN+S3).(A0'+B0'+M'+S3).(A0'+B0'+CN'+M+S3').(A0'+B0+
CN+S2).(A0'+B0+M'+S2).(A0'+B0+CN'+M+S2').(A0+B0'+CN'+M+S1).(A0+B
0+CN+S1').(A0+B0+M'+S1').(A0+B0'+M'+S0').(A0+B0'+CN+S0').(A0+B0
'+CN'+M+S0)
```

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :

```
**************************************************
```

```
(M'+S0'+S1'+S2+S3).(CN+S0'+S1'+S2+S3).(CN'+M+S0+S1+S2'+S3').(A0
'+CN+S2+S3).(A0'+M'+S2+S3).(A0'+CN'+M+S2'+S3').(B0+CN'+M+S1+S2'
).(B0+CN+S1'+S2).(B0+M'+S1'+S2).(B0'+M'+S0'+S3).(A0+M'+S0'+S1')
.(B0'+CN+S0'+S3).(A0+CN+S0'+S1').(B0'+CN'+M+S0+S3').(A0+CN'+M+S
0+S1)
```

. FIG. A0.13

"CIRCUIT REEL"

EXPRESSIONS DE LA SORTIE F_0

********** EQUATION DONNEE EN ENTREE **********

```
      F1=(((B1'.S3.A1+A1.S2.B1')'+(B1'.S1+B1.S0+A1)').((B1.S3.A1+A1.
F101*S2.B1')'.(B1'.S1+B1.S0+A1)')'+(CN.M'.(B0.S3.A0+A0.S2.B0')'+M'
F102*.(B0'.S1+B0.S0+A0)')').(((B1.S3.A1+A1.S2.B1')'+(B1'.S1+B1.S0+
F103*A1)').((B1.S3.A1+A1.S2.B1')'.(B1'.S1+B1.S0+A1)')'.(CN.M'.(B0.
F104*S3.A0+A0.S2.B0')'+M'.(B0'.S1+B0.S0+A0)')')'
```

********** EXPRESSION REDUITE DE L'EQUATION **********

LE RESULTAT EST DONNE EN PREMIERE FORME.

```
F1=B0.B1.CN.M'.S0.S3'+A0.B0.B1.S0'.S3+B1.CN.M'.S0.S2'.S3'+A0'.A
1.B0.B1.M'.S0'.S3'+A0'.A1.B1.CN.M'.S3'+A0.A1'.B0.B1'.S1'.S3+B1'
.CN.M'.S1.S2'.S3'+A1.B0.B1.CN'.S0.S3+A0.A1.S2.S3+A0.A1.B0.B1.S3
+A1.B0.B1.CN.M'.S3'+A1.CN.M'.S2'.S3'+A1'.B0.B1'.CN.M'.S1.S3'+A1
.B1.M.S3+A0.A1.B1.CN'.S3+A0.A1.B0'.B1'.S2+B0'.B1'.CN.M'.S1.S2'+
A1.B0'.B1'.CN'.S1.S2+A1.B0'.B1'.CN.M'.S2'+A0'.A1.B0'.B1'.M'.S1'
.S3'+A0'.A1.B1'.CN.M'.S2'+A1.B1'.M.S2+A1'.B0'.B1.CN.M'.S0.S2'+A
0.A1'.B0'.B1.S0'.S2+A0.A1.B1'.CN'.S2+A0.B0'.B1'.S1'.S2+A1'.B1.M
.S0'+A0.A1'.B1.CN'.S0'+A0.A1'.B1'.CN'.S1'+A0'.A1'.B1'.CN.M'.S1+
A1'.B1'.M.S1'+A1'.B0'.B1.CN'.S0'.S1+A1'.B0.B1'.CN'.S0.S1'+A0'.A
1'.B0'.B1.M'.S0.S1'+A0'.A1'.B0.B1'.M'.S0'.S1+A0'.A1'.B1.CN.M'.S
0+A0'.A1.B0'.B1.M'.S1'.S3'+A0'.A1.B0.B1'.M'.S0'.S2'+A1.B0'.B1.C
N'.S1.S3+A1.B0.B1'.CN'.S0.S2
```

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
****

```
A0.S0'.S1'.S2.S3+A0.B1'.S1'.S2.S3+A1.CN'.S0.S1.S2.S3+A0.B1.S0'.
S2.S3+A1.B0.CN'.S0.S2.S3+A1.B1'.CN'.S0.S1.S2+A1.B1.CN'.S0.S1.S3
+A1.B0'.CN'.S1.S2.S3+B0'.B1.CN'.S0'.S1.S3+B0.B1'.CN'.S0.S1'.S2+
A0'.A1.B0.M'.S0'.S2'.S3'+A0'.A1.M'.S0'.S1'.S2'.S3'+A0'.A1.B1'.M
'.S0'.S1'.S2'+A0'.B0.B1'.M'.S0'.S1.S2'+A0'.A1.B1.M'.S0'.S1'.S3'
+A0'.A1.B0'.M'.S1'.S2'.S3'+A0'.B0'.B1.M'.S0.S1'.S3'+M.S0'.S1'.S
2.S3+A0.A1'.B0.S0'.S1'.S3+CN.M'.S0.S1.S2'.S3'+A0.A1.B0.B1.CN.M'
+A1'.B0.CN.M'.S0.S1.S3'+A0.A1.B0'.B1'.CN.M'+A1.M.S2.S3+A1'.B0'.
CN.M'.S0.S1.S2'+A0.A1'.B0'.S0'.S1'.S2+B1.M.S0'.S3+A0.B1.CN'.S0'
.S2+A0.B1'.CN'.S1'.S2+A0.A1'.CN'.S0'.S1'+A0'.B1'.CN.M'.S1.S2'+B
1'.M.S1'.S2+A1'.M.S0'.S1'+A0'.B1.CN.M'.S0.S3'+A0'.A1'.CN.M'.S0.
S1
```

.FIG_A0.14
'CIRCUIT REEL'
EXPRESSION DE LA SORTIE F_1

********** EQUATION DONNEE EN ENTREE **********

```
F200 F2=(((B2.S3.A2+A2.S2.B2')'+(B2'.S1+B2.S0+A2)').((B2.S3.A2+A2.
F201*S2.B2')'.(B2'.S1+B2.S0+A2)')'+(CN.(B1.S3.A1+A1.S2.B1')'.(B0.S3.
F202*A0+A0.S2.B2')'.M'+M'.(B1.S3.A1+A1.S2.B1')'.(B0'.S1+B0.S0+A0)'
F203*+M'.(B1'.S1+B1.S0+A1)')')'.(((B2.S3.A2+A2.S2.B2')'+(B2'.S1+B2.
F204*S0+A2)')'.((B2.S3.A2+A2.S2.B2')'.(B2'.S1+B2.S0+A2)')'.(CN.(B1.
F205*S3.A1+A1.S2.B1')'.(B0.S3.A0+A0.S2.B0')'.M'+M'.(B1.S3.A1+A1.S2.
F206*B1')'.(B0'.S1+B0.S0+A0)'+M'.(B1'.S1+B1.S0+A1)')')'
```

********** EXPRESSION REDUITE DE L'EQUATION F200****

LE RESULTAT EST DONNE EN PREMIERE FORME.

```
F200 F2=A0'.A2.B0.B1.B2.M'.S0'.S2'+B0.B1.B2.CN.M'.S0.S3'+A0'.B0'.B2.
     M'.S0.S1'.S2'.S3'+A0'.B0'.B1.B2.M'.S0.S1'.S3'+A0'.B1.B2.CN.M'.S
     0.S3'+B2.CN.M'.S0.S2'.S3'+A1'.B0.B2.CN.M'.S0.S3'+A0'.A1'.A2.B0.
     B2.M'.S0'.S3'+A0'.A2.B1.B2.CN.M'.S3'+A2.B0.B1.B2.CN.M'.S3'+A1'.
     A2.B0.B2.CN.M'.S3'+A0'.A1'.A2.B2.CN.M'.S3'+A1.A2.B0.B2.CN'.S0.S
     3+A0'.B0.B2'.M'.S0'.S1.S2'.S3'+B2'.CN.M'.S1.S2'.S3'+A1'.A2.B1.B
     2.M'.S0'.S2'+A0.A1.A2.B2.CN'.S3+A0'.A2.B0'.M'.S1'.S2'.S3'+A0'.A
     2.B0.M'.S0'.S2'.S3'+A0.A2.B1.B2.CN'.S0.S3+A2.B0.B1.B2.CN'.S0.S3
     +A1'.A2'.B0.B2'.CN.M'.S1.S3'+A2.B2.CN'.S0.S1.S3+A2'.B0.B1.B2'.C
     N.M'.S1.S3'+A0'.A2'.B1.B2'.CN.M'.S1.S3'+A0'.A2'.B0.B1.B2'.M'.S0
     '.S1.S3'+A2.CN.M'.S2'.S3'+A0.A1.A2'.B0.B2'.S1'.S3+A2.B2.M.S3+A0
     .A1.B0.B2.S0'.S3+A0.A1.A2.B0.B2.S3+A0.A2'.B0.B1.B2'.S0.S1'.S3+A
     0.A2.B0.B1.B2.S0.S3+A0.B0.B1'.B2.S0'.S1.S3+A1.B1.B2.S0'.S3+A0.A
     2.B1.S0.S2.S3+A0.A2.B1'.S1.S2.S3+A1.A2'.B1.B2'.S1'.S3+A1.A2.B1.
     B2.S3+A1.A2.S2.S3+A0'.A2.B0'.B1'.B2'.M'.S1'.S2'+A0'.A2.B1'.B2'.
     CN.M'.S2'+A1'.A2.B0'.B2'.CN.M'.S2'+A0'.A1'.A2.B2'.CN.M'.S2'+A1'
     .A2.B1'.B2'.M'.S1'.S2'+A0'.B0.B1'.B2'.M'.S0'.S1.S2'+A0'.A2'.B0'
     .B1'.B2.M'.S0.S1'.S2'+B0'.B1'.B2'.CN.M'.S1.S2'+A0.A1.A2.B0'.B2'
     .S2+A0'.B1'.B2'.CN.M'.S1.S2'+A0.B0'.B1.B2'.S0.S1'.S2+A1'.B0'.B2
     '.CN.M'.S1.S2'+A2.B2'.M.S2+A0.A1.A2'.B0'.B2.S0'.S2+A1'.A2'.B0'.
     B2.CN.M'.S0.S2'+A0.A1.B0'.B2'.S1'.S2+A2'.B0'.B1'.B2.CN.M'.S0.S2
     '+A0.A1.A2.B2'.CN'.S2+A2.B2'.CN'.S0.S1.S2+A0.A2'.B0'.B1'.B2.S0'
     .S1.S2+A0.A2.B0'.B1'.B2'.S1.S2+A0'.A2'.B1'.B2.CN.M'.S0.S2'+A0.A
     2.B1'.B2'.CN'.S1.S2+A0'.A1'.A2.B0'.B2'.M'.S1'.S2'+A2.B0'.B1'.B2
     '.CN.M'.S2'+A1.A2'.B1'.B2.S0'.S2+A1.A2.B1'.B2'.S2+A1.B1'.B2'.S1
     '.S2+A2.B0'.B1'.B2'.CN'.S1.S2+A1.A2.B0'.B2'.CN'.S1.S2+A2'.B2.M.
     S0'+A0.A1.A2'.B2.CN'.S0'+A0'.A1'.A2'.B2.CN.M'.S0+A1'.A2'.B1.B2'
     .M'.S0'.S1+A0.A1.A2'.B2'.CN'.S1'+A0.A2'.B1'.B2.CN'.S0'.S1+A0'.A
     1'.A2'.B0'.B2.M'.S0.S1'+A2'.B0'.B1'.B2.CN'.S0'.S1+A0.A2'.B1.B2'
     .CN'.S0.S1'+A2'.B2'.M.S1'+A1.A2'.B0'.B2.CN'.S0'.S1+A2'.B0.B1.B2
     '.CN'.S0.S1'+A0'.A1'.A2'.B2'.CN.M'.S1+A0'.A1'.A2'.B0.B2'.M'.S0'
     .S1+A1.A2'.B0.B2'.CN'.S0.S1'+A1'.A2'.B1'.B2.M'.S0.S1'+A0.A2.B0'
     .B2'.S0.S1.S2+A0.A2.B0.B2.S0.S1.S3+A1'.A2.B1.B2'.M'.S0'.S2'+B0.
     B1.B2'.CN'.S0.S1'.S2+A1'.A2.B1'.B2.M'.S1'.S3'+B0'.B1'.B2.CN'.S0
     '.S1.S3+A0'.A2.B0'.B1.B2.M'.S1'.S3'+A1.B2.S0'.S2.S3+A0.A1'.A2.B
     1'.B1.B2'.CN.M'.S0
```

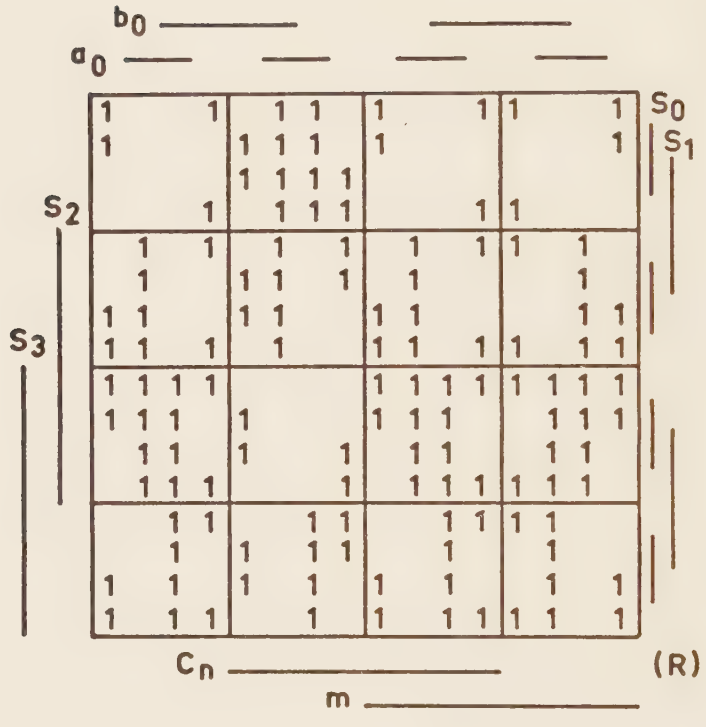
FIG. A0_15
"CIRCUIT RÉEL"
REPRÉSENTATION PREMIÈRE DE LA SORTIE F_2

POUR OBTENIR LA REPRESENTATION COMPLETE, AJOUTER :
****

```
A1.S0'.S1'.S2.S3+A0'.A1'.A2.B0'.B2.M'.S1'.S3'+A2.B0'.B1'.CN'.S1
.S2.S3+A0'.A1'.B0.B2'.M'.S0'.S1.S2'+A0.B1.B2'.S0.S1'.S2.S3+A0.B
1'.B2.S0'.S1.S2.S3+A1.B2'.S1'.S2.S3+A0.A2.B1.B2'.CN'.S0.S2+A0.A
2.B0'.B1.B2'.CN.M'.S0.S3'+A1.A2.B0'.B2.CN'.S1.S3+A0'.A2.B1.B2.M
'.S0'.S1'.S3'+A0'.A1.A2.B0'.B1.B2.M'.S1'+A2.B0'.B1'.B2.CN'.S1.S
3+A0.A2.B1'.B2.CN'.S1.S3+A0.B1'.B2.CN'.S0'.S1.S3+A1.B0'.B2.CN'.
S0'.S1.S3+A1'.A2.M'.S0'.S1'.S2'.S3'+A1'.A2.B2.M'.S0'.S1'.S3'+A1
'.A2.B1'.M'.S1'.S2'.S3'+A0'.A1'.B0'.B2.M'.S0.S1'.S3'+A1'.B1'.B2
.M'.S0.S1'.S3'+A0.B1.B2'.CN'.S0.S1'.S2+A0'.A1.A2.B1.B2.M'.S0'.S
1'+A2.B0.B1.CN'.S0.S2.S3+A1.A2.B0.B2'.CN'.S0.S2+A2.B0.B1.B2'.CN
'.S0.S2+A1.B0.B2'.CN'.S0.S1'.S2+A1'.A2.B1.M'.S0'.S2'.S3'+A1'.A2
.B2'.M'.S0'.S1'.S2'+A1'.B1.B2'.M'.S0'.S1.S2'+A0'.A1'.A2.B0.B2'.
M'.S0'.S2'+A0.A2.B0.B2.CN.M'.S0.S1.S2'+A0.A1'.A2.B0.B2.CN.M'.S0
.S1+A0.A1'.A2.B0.B1'.B2.CN.M'.S1+A0.A2.B0.B1'.B2.CN.M'.S1.S2'+A
0.A2.B0.B1'.B2.S1.S3+A0.A2.B0'.B1.B2'.S0.S2+A0'.A1.A2.B0.B1'.B2
'.M'.S0'+A0'.A2.B0.B1'.B2'.M'.S0'.S2'+A0'.A2.B1'.B2'.M'.S0'.S1'
.S2'+A0'.A1.A2.B1'.B2'.M'.S0'.S1'+A0.A2.B0'.B2'.CN.M'.S0.S1.S3'
+A0.A1'.A2.B1'.B2'.CN.M'.S0.S1+A.S0'.S1'.S2.S3+CN.M'.S0.S1.S2'.
S3'+A0'.A2.M'.S0'.S1'.S2'.S3'+A1'.A2'.B0.CN.M'.S0.S1.S3'+A2'.B0
.B1.CN.M'.S0.S1.S3'+A0'.A2'.B1.CN.M'.S0.S1.S3'+A0.A1.A2'.B0.S0'
.S1'.S3+A0.A1.A2.B0.B2.CN.M'.S2'+A0.A2.B0.B1.B2.CN.M'.S0+A0.A2.
S0.S1.S2.S3+A1.A2'.B1.S0'.S1'.S3+A0'.A1.A2.B0.B1.B2.M'.S0'+A0'.
A1.A2.B1.B2.CN.M'+A1.A2.B0.B1.B2.CN.M'+A1.A2.B1.B2.CN.M'.S2'+A0
.A1.A2.B0'.B2'.CN.M'.S3'+A2.M.S2.S3+A1'.A2'.B0'.CN.M'.S0.S1.S2'
+A0.A1.A2'.B0'.S0'.S1'.S2+A2'.B0'.B1'.CN.M'.S0.S1.S2'+A2.CN'.S0
.S1.S2.S3+A0'.A2'.B1'.CN.M'.S0.S1.S2'+A0.A2.B0'.B1'.B2'.CN.M'.S
1+A1.A2.B1'.B2'.CN.M'.S3'+A0'.A1.A2.B1'.B2'.CN.M'+A1.A2.B0'.B1'
.B2'.CN.M'+A0'.A1.A2.B0'.B1'.B2'.M'.S1'+A1.A2'.B1'.S0'.S1'.S2+B
2.M.S0'.S3+A0.A1.B2.CN'.S0'.S3+A0'.A1'.B2.CN.M'.S0.S3'+A0.A1.B2
'.CN'.S1'.S2+A0.A1.A2'.CN'.S0'.S1'+B2'.M.S1'.S2+A2'.M.S0'.S1'+A
0'.A1'.B2'.CN.M'.S1.S2'+A0'.A1'.A2'.CN.M'.S0.S1
```

FIG. A0_16
"CIRCUIT RÉEL"
SORTIE F_2
IMPLICANTS PREMIERS INESSENTIELS OU PSEUDO_INESSENTIELS

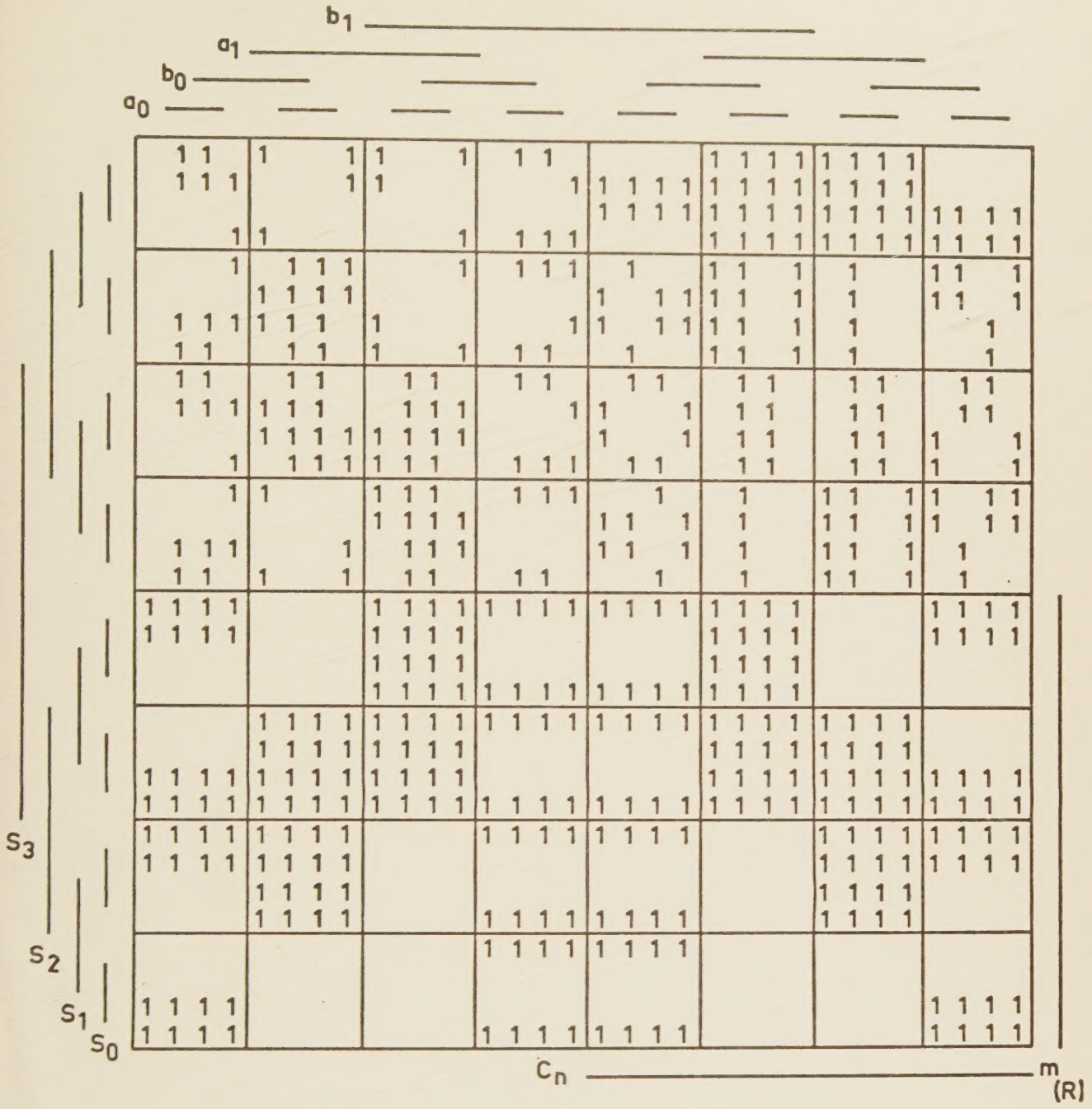
PROGRAMME CHOPIN
ANNEXE 0
EXEMPLES



F_0

FIGURE A0_17
CIRCUIT REEL
SORTIE F0
DIAGRAMME A KARNAUGH

PROGRAMME CHOPIN
ANNEXE 0
EXEMPLE



-F1-
FIGURE A0-18
CIRCUIT REEL
SORTIE F1
DIAGRAMME A KARNAUGH